Universidade Estadual Paulista “Júlio de Mesquita Filho”
Faculdade de Engenharia – Câmpus de Bauru
Programa de Pós-graduação em Engenharia de Produção

Heurísticas para o floorplanning de circuitos VLSI como um problema de empacotamento de retângulos flexíveis

Letícia Leite Pavanello

Bauru
2022

Letícia Leite Pavanello

Heurísticas para o floorplanning de circuitos VLSI como um problema de empacotamento de retângulos flexíveis

Dissertação apresentada ao Curso de Pós-graduação em Engenharia de Produção da Universidade Estadual Paulista "Júlio de Mesquita Filho" como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia de Produção.

Orientador: Profa. Dra. Adriana Cristina Cherri

Coorientador: Prof. Dr. Carlos Diego Rodrigues

Bauru
2022

P337h

Pavanello, Leticia Leite

Heurísticas para o floorplanning de circuitos VLSI como um problema de empacotamento de retângulos flexíveis / Leticia Leite Pavanello. -- Bauru, 2022

56 f.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, Bauru

Orientadora: Adriana Cristina Cherri

Coorientador: Carlos Diego Rodrigues

1. Pesquisa Operacional. 2. Problema de Empacotamento. 3. Retângulos flexíveis. 4. Matheurística. 5. BRKGA. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE LETÍCIA LEITE PAVANELLO, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO, DA FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.

Aos 27 dias do mês de outubro do ano de 2022, às 14:00 horas, por meio de Videoconferência, realizou-se a defesa de DISSERTAÇÃO DE MESTRADO de LETÍCIA LEITE PAVANELLO, intitulada **Heurísticas para o floorplanning de circuitos VLSI como um problema de empacotamento de retângulos flexíveis**. A Comissão Examinadora foi constituida pelos seguintes membros: Profª. Drª. ADRIANA CRISTINA CHERRI NICOLA (Orientador(a) - Participação Virtual) do(a) Departamento de Matematica / Faculdade de Ciencias de Bauru UNESP, Prof. Dr. FLAVIO KEIDI MIYAZAWA (Participação Virtual) do(a) Instituto de Computação / Universidade Estadual de Campinas, Prof. Dr. PEDRO AUGUSTO MUNARI JUNIOR (Participação Virtual) do(a) Departamento de Engenharia de Produção / Universidade Federal de São Carlos. Após a exposição pela mestranda e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, a discente recebeu o conceito final: aprovada . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Profª. Drª. ADRIANA CRISTINA CHERRI NICOLA

À minha mãe Márcia e minha irmã Alícia, testemunhas da minha dedicação...

Agradecimentos

Agradeço à Deus, por ter sempre me sustentado, principalmente em meio ao contexto desafiador de pandemia, no qual este trabalho foi realizado.

Aos meus orientadores Adriana Cherri e Diego Rodrigues, pela orientação, amizade, confiança e todo o suporte dado durante o desenvolvimento deste trabalho.

À minha mãe Márcia, pelo amor incondicional, cuidado, incentivo e dedicação. À minha irmã Alícia, pelo amor, compreensão e carinho. Obrigada por estarem sempre ao meu lado.

Ao meu querido Raphael Bueno, pelo amor, apoio e todos os momentos que me fizeram mais feliz e motivada. Fico feliz em compartilhar esta etapa da minha vida com você.

Aos meus amigos Arthur Medeiros e Guilherme Calvo, pela amizade, apoio e companhia nos mais diversos momentos.

Aos professores Pedro Munari e Flávio Miyazawa pelas sugestões e dicas que enriqueceram esta dissertação.

À CAPES pela credibilidade e apoio financeiro.

Agradeço a todos que me ajudaram, direta ou indiretamente, a concluir esta etapa.

Para sempre, obrigada.

Resumo

O problema de planejamento de *layout* de um circuito VLSI *(Very Large Scale Integration)*, chamado de *floorplanning*, consiste no processo de determinar a localização física de módulos retangulares e interconectá-los dentro dos limites do chip, otimizando recursos. Com forte característica geométrica, este problema pode ser abordado como um problema de empacotamento de retângulos flexíveis. Neste problema, é preciso definir as posições de módulos em uma área de alocação sem que haja sobreposição, além de decidir as dimensões de cada módulo, que podem ser ajustadas dentro de uma proporção predefinida, e de modo a minimizar o comprimento de fio utilizado para conectar os módulos entre si. Devido ao grande número de variáveis envolvidas, este problema é difícil de ser solucionado e, obter soluções exatas, implica em alta complexidade computacional. Desta forma, métodos heurísticos são comumente utilizados na tentativa de obter boas soluções em tempos viáveis. Para resolver o problema de planejamento de circuitos VLSI de maneira eficiente, um modelo matemático, uma abordagem matheurística e uma meta-heurística BRKGA (*Biased Random-Key Genetic Algorithm*) são propostos neste trabalho. O modelo matemático é utilizado em ambas as abordagens heurísticas de forma iterativa e com uma estratégia de janela deslizante, resolvendo o problema parcialmente para reduzir a dificuldade computacional do problema original. O desempenho dos métodos propostos foi avaliado a partir de testes computacionais com instâncias MCNC (*Microelectronics Center of North Carolina*) na linguagem de programação C++ com o solver CPLEX. Resultados dos experimentos computacionais demonstraram grande potencial de obtenção de soluções satisfatórias pelos métodos, principalmente na abordagem BRKGA combinada com o procedimento de janela deslizante.

Palavras-chave: Problema de Empacotamento. Retângulos flexíveis. Matheurística. BRKGA.

Abstract

The floorplanning problem of a VLSI (Very Large Scale Integration) circuit is the process of determining the physical location of rectangular modules and interconnecting them within the chip limits, optimizing resources. With strong geometric characteristics, this problem can be approached as a flexible rectangle packing problem. In this problem, it is necessary to define the positions of modules in an allocation area without overlapping, in addition to deciding the dimensions of each module, which can be adjusted within a predefined proportion, in order to minimize the wire length used for connecting the modules together. Due to the large number of variables involved, this problem is difficult to solve, and obtaining exact solutions implies high computational complexity. In this way, heuristic methods are commonly used in an attempt to obtain good solutions in feasible times. To efficiently solve the VLSI circuit planning problem, a mathematical model, a matheuristic approach, and a BRKGA (Biased Random-Key Genetic Algorithm) are proposed in this work. The mathematical model is used in both heuristic approaches in an iterative way and with a sliding window strategy, solving the problem partially to reduce the computational difficulty of the original problem. The performance of the proposed methods was evaluated from computational tests with MCNC instances (Microelectronics Center of North Carolina) in the C++ programming language with the CPLEX solver. Results of computational experiments showed great potential for obtaining satisfactory solutions by the methods, mainly in the BRKGA approach combined with the sliding window procedure.

Keywords: Floorplanning VLSI. Packaging Problem. Soft rectangles. Matheuristics. BRKGA.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

BRKGA	*Biased Random-Key Genetic Algorithm*
BRKGA-SW	*BRKGA - Sliding Window*
BRKGA-CLP	*BRKGA - single Container Loading Problem*
GAOSPF	*Genetic Algorithm for Open Shortest Path First routing*
HSA	*Hybrid Simulated Annealing*
HPWL	*Half Perimeter Wire Length*
LOA	*Lion Optimization Algorithm*
MCNC	*Microelectronics Center of North Carolina*
RKGA	*Random-Key Genetic Algorithm*
PCE	*Problemas de Corte e Empacotamento*
SW	*Sliding Window*
VLSI	*Very Large Scale Integration*

Sumário

1 Introdução

O VLSI (*Very Large Scale Integration*) é uma tecnologia de microeletrônica que integra uma grande quantidade de módulos em um chip. A construção de um circuito integrado é feita de forma sequencial, devido à sua complexidade, e o *floorplanning*, processo de determinar a localização física dos módulos e interconectá-los dentro dos limites do chip do melhor modo possível, é uma etapa essencial no projeto físico de um VLSI. Nas abordagens modernas deste processo, os módulos são essencialmente flexíveis (CHEN; CHANG, 2009; SINGH; BAGHEL; AGARWAL, 2016), assim as dimensões de cada módulo pode ser ajustada dentro de um intervalo de proporção entre altura e largura pré-definida no projeto.

O *floorplanning* permite que a complexidade do projeto do circuito seja gerenciada de modo eficaz, dado que a planta de alocação resultante (*layout*) afeta todas os estágios subsequentes no projeto, como colocação e roteamento. Este planejamento envolve determinar os locais e tamanhos dos módulos no chip, além de estimar a área total necessária, o atraso e o congestionamento da fiação, fornecendo, assim, uma base para o *layout*. Este processo de planejamento físico de um circuito pode ser interpretado como um problema de empacotamento com retângulos flexíveis. Computacionalmente, ele é um problema NP-difícil (KAHNG et al., 2011).

Os problemas de empacotamento são estudados há décadas como uma ferramenta de apoio na tomada de decisão em diversos setores. Com uma forte característica geométrica, os problemas de empacotamento bidimensionais consistem em empacotar retângulos menores (itens) com dimensões predefinidas em regiões retangulares maiores, utilizando os recursos disponíveis da melhor forma possível e sem que haja sobreposição. Apesar de ser um problema difícil de otimização combinatória, diversos trabalhos foram publicados. Abordagens exatas para o problema foram propostas por Fekete e Schepers (2000), Chen e Chang (2009) e Martello e Monaci (2015). Entretanto, devido à resolução custosa deste problema, muitas abordagens de resolução são heurísticas. A heurística mais antiga e famosa é a *bottom-left* (BL), proposta por Baker, Coffman Jr e Rivest (1980). Outras abordagens foram propostas por Wu et al. (2002), Huang, Chen e Xu (2007), Wei, Zhang e Chen (2009), Wei et al. (2011), Bortfeldt (2013), He, Ji e Li (2015), Wang e Chen (2015), Venkatraman e Sundhararajan (2019), entre outros.

Uma variação para o problema de empacotamento bidimensional ocorre quando os retângulos a serem empacotados possuem dimensões flexíveis. Neste caso, as dimensões de cada item podem variar em um determinado intervalo, respeitando uma proporção entre suas

respectivas alturas e larguras. Os problemas de empacotamento de retângulos flexíveis são problemas de otimização combinatória difíceis de serem resolvidos, pois é necessário definir como os retângulos de um conjunto predefinido serão empacotados na região e, ainda, decidir quais as suas dimensões, respeitando as suas limitações geométricas e a não-sobreposição dos itens. Além de sua importância teórica, este problema surge em várias situações práticas, incluindo a determinação do *layout* de circuitos integrados de grande escala.

Os problemas de empacotamento de módulos (retângulos) flexíveis em circuitos VLSI costumam considerar a otimização da área do chip e o comprimento do fio de metal semi-condutor, usado para conectar os módulos entre si. Para minimizar o comprimento do fio das interconexões do circuito, é preciso identificar quais itens devem ser colocados próximos uns aos outros e alocá-los de forma a atender metas, muitas vezes conflitantes, de comprimento de fio utilizado, espaço disponível no chip e desempenho necessário. Com os módulos posicionados de acordo com uma descrição detalhada de conectividade do circuito, obtém-se distâncias de interconexão mais curtas, que utilizam menos recursos de roteamento, e consequentemente circuitos mais eficientes.

Devido ao grande número de variáveis envolvidas no problema, o *floorplanning* é difícil de ser solucionado, apesar de ser representado facilmente através de modelos matemáticos. Consequentemente, obter soluções exatas torna-se inviável, principalmente para grandes instâncias. Desta forma, métodos heurísticos podem ser eficientes para encontrar boas soluções em tempos computacionais viáveis. Em Chen, Zhu e Ali (2011), Anand, Saravanasankar e Subbaraj (2013), Chan, Hsu e Lin (2013), He, Ji e Li (2015), Wu et al. (2016), Venkatraman e Sundhararajan (2019) e Shunmugathammal, Columbus e Anand (2020), os autores sugeriram e exploraram diferentes técnicas heurísticas e meta-heurísticas para o *floorplanning*, sendo o algoritmo genético utilizado em várias das abordagens.

O algoritmo genético é uma meta-heurística em que cada indivíduo da população representa uma solução viável para o problema abordado. Através de operações de cruzamento e mutação, os indivíduos encontrados sofrem modificações com o objetivo de evoluir a população e possivelmente obter soluções melhores a cada iteração. O BRKGA (*Biased Random-Key Genetic Algorithm*) é um algoritmo genético de chave tendenciosa amplamente utilizado em problemas de natureza combinatória, devido à sua estrutura de fácil adaptação. Os trabalhos Ericsson, Resende e Pardalos (2002), Gonçalves, Mendes e Resende (2005), Noronha, Resende e Ribeiro (2011), Mainieri (2014), Mönch e Roob (2018), Mauri et al. (2021), entre outros, apresentaram o BRKGA, aplicados em diferentes contextos e obtendo bons resultados. Não foram encontrados trabalhos de BRKGA para empacotar retângulos flexíveis ou aplicado ao *floorplanning* de circuitos.

Apesar dos trabalhos existentes e dos diversos métodos que abordam o *floorplanning* de circuitos VLSI, as soluções ótimas não são conhecidas e abordagens não-exatas mais eficientes podem ser exploradas. Ainda, a maioria dos autores adota como objetivo principal a minimização da área do chip, deixando o comprimento de fio como objetivo secundário, quando considerado.

Devido à sua importância para garantia de qualidade e eficiência em um circuito VLSI, o comprimento de fio deve ser minimizado no processo de alocação e, com distâncias mais curtas para as interconexões, é esperado que a área do chip seja consequentemente reduzida.

Neste trabalho, o *floorplanning* é abordado como um problema de empacotamento de retângulos flexíveis com o objetivo de minimizar o comprimento de fio utilizado nas interconexões. Para resolver o problema de planejamento de circuitos VLSI de maneira eficiente, são propostos um modelo matemático, uma heurística de janela deslizante e uma meta-heurística BRKGA adaptada. Simultaneamente são exploradas as vantagens da programação matemática e do método heurístico. Para ambos os procedimentos propostos, estratégias de inicialização e de janela deslizante são definidas. Desta forma, a resolução do problema de alocação é realizada de forma parcial em relação ao problema original. Ao determinar o tamanho da janela deslizante, uma quantidade menor e fixa de variáveis com posicionamento livre são consideradas, diminuindo significativamente a dificuldade envolvida na resolução modelo.

Além da característica de flexibilidade dos itens, a influência de diferentes tamanhos de janela deslizante e critérios na ordenação inicial sobre o resultado final das abordagens é analisada. A implementação dos métodos foi realizada na linguagem computacional C++ com o solver IBM CPLEX Optimization Studio versão 22.1. Os testes foram realizados com instâncias da literatura e *layouts* de circuitos VLSI com valores de comprimento de fio em diferentes tempos computacionais são apresentados. Uma comparação entre os resultados dos métodos propostos é realizada.

O Capítulo 2 deste trabalho é composto pela fundamentação teórica para o estudo desenvolvido e o Capítulo 3 contém uma revisão de literatura que contempla problemas de empacotamento em geral e, principalmente, abordados para o resolver o *floorplanning*, além de métodos heurísticos conhecidos. Conceitos fundamentais e o modelo matemático proposto são detalhados no Capítulo 4, seguido do procedimento heurístico desenvolvido e algoritmo genético adaptado (BRKGA-SW) apresentados no Capítulo 5. No Capítulo 6, resultados de testes computacionais com instâncias da literatura são apresentados, e o Capítulo 7 destina-se às conclusões e melhorias sugeridas para a continuidade do trabalho.

2 Fundamentação Teórica

Neste capítulo, são apresentadas definições detalhadas que fundamentam o *floorplanning* de circuitos VLSI e os problemas de empacotamento de retângulos. Além disso, são mencionados possíveis métodos de resolução propostos ao longo deste trabalho.

2.1 Problemas de planejamento de circuitos VLSI

O VLSI é uma tecnologia de microeletrônica que integra uma grande quantidade de módulos eletrônicos numa pastilha de silício, conhecida como chip. Com o avanço da tecnologia de circuito integrado e a sua presença crescente na composição de produtos de uso específico, tais como automóveis, eletrodomésticos e dispositivos de comunicação em geral, torna-se necessário planejar detalhadamente cada etapa do projeto de fabricação, geralmente dividas em duas fases: projeto lógico e projeto físico. A Figura 1 apresenta um circuito integrado VLSI.

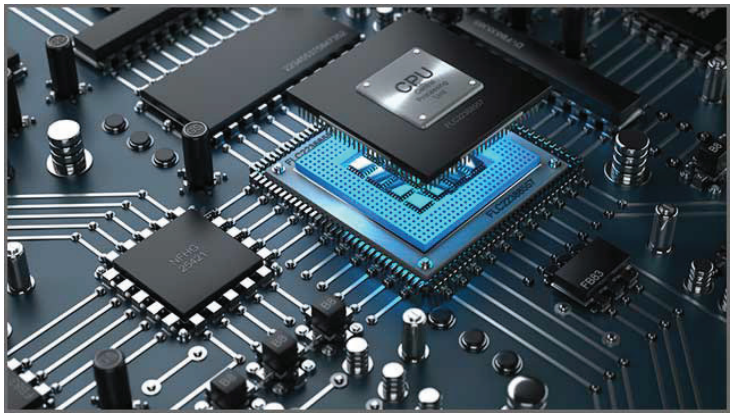


Figura 1 – Circuito integrado VLSI.

Fonte: Iaroslav Neliubov/Shutterstock

O projeto físico é essencial no VLSI. Nesta etapa, as representações de circuito dos módulos do projeto são convertidas em representações geométricas de forma que, quando fabricadas nas camadas de materiais correspondentes, garantirão o funcionamento necessário do circuito. Os principais passos que compõe o projeto físico são o planejamento topológico, posicionamento e roteamento, assim um conjunto de módulos e um *netlist*, isto é, uma descrição detalhada de conectividade do circuito, são determinados para serem posicionados e conectados no chip.

Um dos grandes desafios no design de um VLSI é o problema de planejamento de *layout* (*floorplanning*), isto é, o processo de determinar a localização física dos módulos e interconectá-los dentro dos limites da placa (chip), utilizando os recursos disponíveis da melhor forma possível. Um *layout*, também chamado de planta baixa ou *floorplan*, é uma dissecção de uma região retilínea em um conjunto de retângulos, em que cada um destes retângulos representa um módulo.

Para o *floorplanning* de circuitos VLSI é dado um conjunto de módulos retangulares e algumas redes, sendo cada rede um subconjunto de módulos que deverão ser conectados por fios semicondutores no passo de roteamento. É chamado de comprimento de fio a distância utilizada para interconectar os módulos que pertencem a uma mesma rede nos circuitos integrados. Conectar os módulos com longas distâncias de comprimento de fio resulta em chips maiores, mais lentos e com menor confiabilidade. Ainda, quando a medida do comprimento total de todas as interconexões é muito grande ou quando as conexões são muito densas em uma determinada região, pode não haver recursos de roteamento suficientes, afetando o custo de fabricação.

Deste modo, busca-se determinar o posicionamento dos módulos de forma a minimizar o comprimento de fio utilizado nas conexões para obter circuitos com maior nível de desempenho. Com módulos posicionados de acordo com a descrição de conectividade do circuito, detalhada na *netlist*, obtém-se distâncias de interconexões mais curtas, que utilizam menos recursos, com os caminhos de sinal ponta a ponta mais rápidos e tempos de rota mais consistentes.

Ainda, dado o aumento da complexidade do projeto e as novas propriedades e requisitos do circuito, o *floorplanning* tem sido abordado de modo reformulado. As novas considerações e desafios tornam o problema mais difícil. Ao contrário do *floorplanning* clássico, que geralmente lida apenas com a área dos módulos para minimizar a região do chip, o *floorplanning* moderno é formulado considerando como crucial a possibilidade de redimensionar as dimensões dos módulos retangulares dentro de um intervalo de proporção e considerando restrições de contorno fixo (CHEN; CHANG, 2009). Os limites da proporção são pré-definidos para cada módulo.

O *floorplanning* moderno do VLSI pode ser modelado como um problema de empacotamento com retângulos flexíveis de grande porte. Neste problema, os módulos são flexíveis e devem ser alocados e conectados uns aos outros respeitando restrições de espaço. Além de determinar as posições dos módulos, suas dimensões também são decisões que devem ser consideradas, minimizando o comprimento do fio usado em suas interconexões. Desta forma, acrescenta-se ao problema de empacotamento de retângulos flexíveis, o problema de minimizar o comprimento de fio utilizado para interconectar os módulos.

Posicionar os módulos nos circuitos VLSI é uma tarefa complexa, assim como a implementação de algoritmos que produzam boas soluções para o problema, em tempo de execução viável. O *floorplanning* é um problema difícil de ser solucionado, devido ao grande número de variáveis envolvidas. Computacionalmente, é um problema NP-difícil (KAHNG et al., 2011).

2.2 Problemas de empacotamento de retângulos

Os problemas de empacotamento de retângulos consistem em encontrar uma configuração para alocar retângulos menores (itens) em retângulos maiores (objetos ou recipientes), de forma que os itens estejam completamente contidos no recipiente e sem sobreposições entre si. O objetivo deste problema é posicionar os itens no recipiente do melhor modo possível, otimizando os espaços ou materiais disponíveis. Na Figura 2 é representada uma solução possível de um problema de empacotamento que contém itens retangulares e cujo objetivo é minimizar a área do retângulo exterior envolvente que contém os itens disponíveis para o posicionamento.

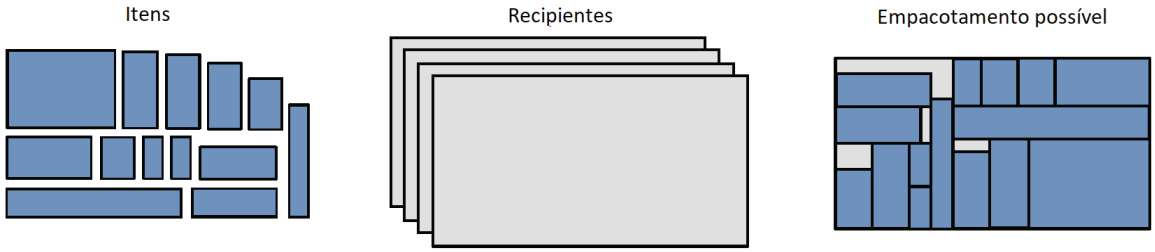


Figura 2 – Representação de um problema de empacotamento de retângulos.

Os problemas de empacotamento são similares ao problema de corte de estoque bidimensional (BEZERRA, 2018). Enquanto o problema de empacotamento busca determinar a melhor configuração para posicionar os itens, o problema de corte consiste em determinar a melhor maneira de produzir os itens através de cortes efetuados nos objetos (placas). Assim, empacotar itens pode ser visto como cortes de espaços, sendo estes espaços ocupados por itens, e vice-versa. Além de sua importância teórica, os Problemas de Corte e Empacotamento (PCE) estão presentes em diversas situações práticas em processos industriais de planejamento de produção, armazenagem e transporte.

2.2.1 Característica do problema

Para especificar um PCE dentre os vários contextos nos quais ele pode surgir, é estabelecida uma tipologia utilizada na literatura para classificar a estrutura de um problema e suas variações (WÄSCHER; HAUβNER; SCHUMANN, 2007). Esta estrutura inclui características suficientes para definir um problema de empacotamento, sendo elas o tipo de atribuição, tipo e forma de itens, tipo de objetos e os critérios de dimensionalidade.

Um PCE pode ser unidimensional, bidimensional, tridimensional ou n-dimensional. O tipo de atribuição de um problema de empacotamento define o objetivo de otimização adotado no posicionamento dos itens no objeto, que pode ser a minimização dos recursos disponíveis ou a maximização da produção, por exemplo. Os itens podem possuir dimensões fixas ou flexíveis, variando altura e largura de modo que a área seja preservada ou com altura e largura que variam

de modo a respeitar uma proporção pré-definida entre elas. Ainda, cada item pode ser regular ou irregular, com posicionamento ortogonal ou não-ortogonal. Sobre os objetos, um problema pode considerar um único objeto, múltiplos objetos idênticos ou múltiplos objetos heterogêneos, disponíveis para alocar os itens.

A forma como um conjunto de itens é cortado ou empacotado no objeto pode ser chamada de padrão de corte, assim a solução de um PCE é representada por uma sequência de padrões que resultam em um *layout* sem sobreposições. Além das características sobre os itens e objetos, a categoria dos padrões de cortes possíveis deve ser considerada para definir o problema. Um padrão pode ser categorizado em guilhotinado ou não-guilhotinado. O corte guilhotinado é feito por toda a extensão do objeto, paralelamente à um dos lados, produzindo dois novos retângulos, isto é, o corte vai de um extremo ao outro do retângulo original. Um padrão é chamado de guilhotinado quando é formado por sucessivos cortes guilhotinados. Caso contrário, assume-se que o padrão é não-guilhotinado.

Com o objetivo de minimizar recursos durante o processo de planejamento de *layout* de circuitos VLSI, o problema de empacotamento ortogonal bidimensional com retângulos flexíveis, alocados em um único objeto de modo não-guilhotinado, é considerado neste trabalho.

2.2.2 Retângulos flexíveis

Uma variação para o problema de empacotamento bidimensional ocorre quando os retângulos a serem empacotados possuem dimensões flexíveis (*soft rectangles*), conforme supracitado. Nesse caso, uma região retangular com comprimento e altura pré-definidos deve ser preenchida por um conjunto de retângulos menores com dimensões flexíveis, que podem variar em um intervalo, respeitando a proporção (*aspect ratio*) entre suas respectivas alturas e larguras. Selecionar quais retângulos flexíveis de um conjunto pré-definido serão empacotados na região e quais suas dimensões, respeitando as suas limitações geométricas e a não-sobreposição dos itens escolhidos, consiste em um desafio adicional na resolução deste problema.

Na Figura 3, as dimensões de um item i são ilustradas, assim como as dimensões x e y da área de alocação. Cada item pode variar mantendo sua área ou variar livremente dentro de um intervalo pré-definido de proporção entre sua altura e largura. Para o objeto (chip), é possível determinar dimensões fixas ou considerar como dimensão a altura e largura do menor retângulo envolvente que contém todos os itens empacotados. Assim, as dimensões do objeto podem ser alteradas durante o processo de posicionamento dos itens.

Para exemplificar o problema de empacotamento com retângulos flexíveis em área fixa, considere uma região retangular inicial com dimensões 140×140. Três modos possíveis de alocação são apresentados na Figura 4. Retângulos com as mesmas cores entre as figuras, possuem a mesma área. Nos três casos, as dimensões dos retângulos alocados variam sem que área seja alterada. É possível notar que a alocação da região no segundo recipiente ficou mais concentrada

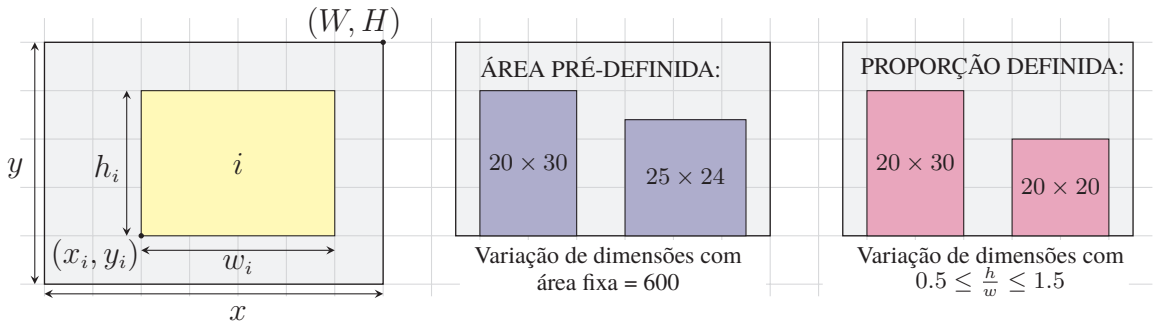


Figura 3 – Dimensões de um item i com exemplos e possíveis variações de dimensão, mantendo a área inicial ou respeitando uma proporção pré-definida.

e, consequentemente, com melhor aproveitamento da região. Ainda, as duas primeiras situações apresenta objetos com dimensões fixas, enquanto a terceira situações assume, como dimensões do objeto, o menor retângulo envolvente que contém todos os itens alocados.

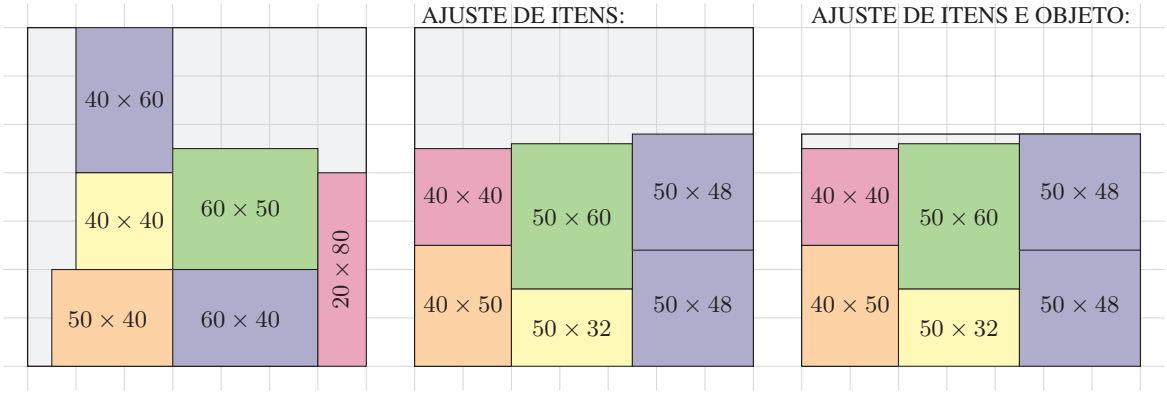


Figura 4 – Representação de possíveis empacotamentos com retângulos flexíveis.

Este problema apresenta inúmeras aplicações, sendo uma das principais, o problema de planejamento e alocação de módulos flexíveis para conexões entre redes em circuitos integrados VLSI (Seção 2.1). Nesta aplicação, cada item, também chamados de módulo, pode ser ajustado de diferentes formas, variando suas dimensões dentro de um intervalo definido de proporção entre altura e largura. Os módulos devem ser alocados em um chip que, neste trabalho, assume as dimensões do menor retângulo envolvente possível.

2.2.3 Dificuldade de resolução

Os PCE pertencem à classe de problema NP-difícil, isto é, não se conhece algoritmo exato que resolva estes problemas em tempo polinomial. Devido sua característica combinatória, existe um número exorbitante de arranjos possíveis quando o objetivo é determinar o posicionamento ótimo de corte ou empacotamento, inviabilizando o processo. Assim, o uso de métodos exatos para otimização de um problema de empacotamento é altamente custoso, pois exige um longo tempo de processamento, sem que sequer sejam gerados resultados práticos.

Com o intuito de viabilizar o processo de resolução de PCE, métodos heurísticos são comumente utilizados para solucionar o problema de empacotamento em tempos computacionais razoáveis. Tais métodos não garantem a obtenção de uma solução ótima, mas podem obter boas aproximações com baixo esforço computacional, isto é, menor tempo de processamento, quando comparado à utilização de métodos exatos. Diferentes métodos foram propostos na literatura para resolver, de modo exato ou aproximado, vários problemas de empacotamento e podem ser agrupados, de forma simplificada, em heurísticas, meta-heurísticas e métodos exatos. Uma breve revisão sobre cada método de resolução é apresentada no Capítulo 3.

3 Revisão de literatura

O problema abordado neste trabalho trata o *floorplanning* de circuitos VLSI como um problema de empacotamento de retângulos flexíveis. Neste capítulo, é apresentada uma revisão de literatura que contempla trabalhos relevantes sobre problemas de empacotamento de modo geral, abordados por meio de diferentes técnicas de resolução, e trabalhos com métodos aplicados especificamente no contexto de planejamento de circuitos integrados.

3.1 Problemas de empacotamento

A literatura sobre problemas de corte e empacotamento ortogonal bidimensional tem crescido consideravelmente nos últimos anos (IORI et al., 2021). Nesta seção são descritos modelos matemáticos propostos para resolver problemas de empacotamento em diversas aplicações, além de heurísticas e meta-heurísticas, comumente utilizadas para obter soluções em tempo computacional viável.

3.1.1 Modelos matemáticos

Para empacotar um conjunto de itens retangulares mistos flexíveis, Murata e Kuh (1998) propuseram um modelo matemático que trata as variáveis de posição relativa dos itens como dados de entrada. A abordagem dos autores é baseada na representação de posicionamento dos itens por pares de sequência (*Sequence Pair*). Um par de sequência é um par de permutações entre dois retângulos e especifica qual das condições de posição é válida: é possível que um módulo esteja à direita ou à esquerda de outro módulo, horizontalmente posicionado, ou que um módulo esteja acima ou abaixo de outro módulo, verticalmente posicionado. A representação adotada pelos autores é a mais utilizada para representar padrões ao longo do tempo.

Em Lodi e Monaci (2003) foi abordado um modelo que executa o empacotamento bidimensional de itens formando níveis. Os itens são inseridos da esquerda para a direita no primeiro nível, que tem sua altura determinada pela altura do item mais alto, e tem como objetivo minimizar a altura utilizada no retângulo envolvente da alocação. Ibaraki e Nakamura (2006) abordaram o problema de empacotamento de retângulos, cujas formas poderiam ser ajustáveis, dentro de determinadas restrições de perímetro e área. Os autores utilizaram pares de sequência

para especificar posições relativas entre cada par de retângulos. Para determinar o tamanho exato e a localização de todos os retângulos, uma formulação matemática foi proposta e para encontrar bons pares de sequências, técnicas de busca local foram utilizadas.

Considerando características de flexibilidade, Bui, Vidal e Hà (2019) propuseram um modelo de programação inteira mista para representar o problema de corte guilhotinado dois-estágios quando os retângulos a serem cortados possuem área fixa, porém, perímetro flexível. Uma abordagem baseada em busca binária também foi proposta. O estudo desenvolvido teve como referência a reforma agrária no Vietnã.

3.1.2 Abordagens heurísticas e meta-heurísticas

Heurísticas são métodos baseados em argumentos intuitivos que geram boas soluções sob certas condições, mas que não garantem a otimalidade. Meta-heurísticas são métodos de otimização flexíveis com estrutura de algorítimos genéricos, que coordenam heurísticas simples e podem ser aplicados a diferentes tipos de problemas com relativamente poucas modificações a serem feitas para adaptá-los (CHERRI, 2009). Muitas meta-heurísticas foram introduzidas na literatura. Entre estas, encontramos *simulated annealing* (recozimento simulado), busca de vizinhança variável, busca de dispersão, busca local iterada, otimização de colônia de formigas, otimização de enxame e algoritmos genéticos.

Imahori, Yagiura e Ibaraki (2003) propuseram meta-heurísticas de busca local com geração de várias soluções iniciais, busca local iterativa e busca aleatória, para otimizar os custos de um empacotamento com retângulos rígidos. Em Imahori, Yagiura e Ibaraki (2005) os autores desenvolveram técnicas para avaliar soluções obtidas por meio da representação de pares de sequência.

Bortfeldt (2006) sugeriu um algoritmo genético para resolver o problema de minimização da área no empacotamento de retângulos, abordando o empacotamento em tiras (*strip packing*) bidimensionais, que consiste em alocar um conjunto de itens retangulares de dimensões fixas em um retângulo maior de largura fixa e largura variável, de modo que a largura total do *layout* seja minimizada. Em Bortfeldt (2013), o mesmo autor apresentou uma abordagem genérica que é baseada em dois algoritmos: uma heurística construtiva e um algoritmo que alterna método de busca e algoritmo genético com restrições que consideram padrões guilhotináveis.

Existe uma vasta literatura sobre heurísticas para problemas de empacotamento bidimensional. Em Ortmann e Vuuren (2010) os autores abordaram dois problemas de empacotamento, o problema de empacotamento em tiras e o problema de empacotamento em caixas de tamanho variável. Além disso, compararam computacionalmente um total de 252 heurísticas e variantes para problemas de empacotamento em tiras bidimensionais.

Em Iori et al. (2021), os autores revisam mais de 180 artigos relacionados a problemas de corte e empacotamento ortogonal bidimensional, com cerca de metade das referências trabalhos

publicados nos últimos dez anos e apenas 20% antes de 2000. Na revisão proposta, foram descritos os principais métodos de pré-processamento e relaxamento, além de alguns algoritmos heurísticos e de aproximação. Os autores discutiram modelos matemáticos e apresentaram estatísticas sobre os métodos exatos para problemas bidimensionais considerados.

3.1.3 Algoritmo genético BRKGA

O algoritmo genético é uma meta-heurística introduzida por Holland (1975) aplicando o princípio de seleção natural de Darwin. Em um algoritmo genético cada indivíduo da população representa uma solução viável para o problema abordado e, dada uma série de gerações produzidas pelo algoritmo, o indivíduo mais apto da última geração é adotado como solução. Em geral, dois tipos de operadores são utilizados: o cruzamento e a mutação.

O cruzamento (*crossover*) é uma operação realizada para definir novos indivíduos que tenham as características genéticas de seus pais. Essa operação intensifica a busca ao procurar obter uma boa solução através da combinação de duas soluções da população. A mutação é uma operação utilizada para perturbar a convergência de uma população e diversificar a busca da solução, procurando gerar indivíduos distintos dos demais indivíduos da população através de uma modificação aleatória. Geralmente, a taxa de mutação é baixa para evitar discrepâncias.

Algoritmos genéticos com chaves aleatórias (RKGA, *Random-Key Genetic Algorithm*) foram propostos pela primeira vez por Bean (1994) para resolver problemas de otimização combinatória envolvendo sequenciamento. Em um RKGA, uma população é representada por vetores de chaves reais geradas aleatoriamente no intervalo contínuo [0,1), sendo chamada de geração a população desenvolvida a cada iteração.

Um algoritmo determinístico, chamado de decodificador, recebe um vetor de chaves aleatórias como entrada e gera uma solução viável do problema de otimização com valor associado da função objetivo. Com esses valores calculados pelo decodificador na geração atual, a população é dividida em dois grupos de indivíduos: um pequeno grupo de elite com os melhores resultados obtidos e o conjunto restante de indivíduos não-elite. Para evoluir a população, uma nova geração de indivíduos deve ser produzida.

Uma nova geração é composta pelos indivíduos de elite da geração anterior, novos vetores gerados aleatoriamente na operação de mutação e indivíduos descendentes, produzidos através do processo de cruzamento de quaisquer dois indivíduos da população. Os descendentes inseridos correspondem sempre às soluções discretas do problema de otimização combinatória, com viabilidade garantida pelo decodificador.

O algoritmo genético de chave aleatória tendenciosa (BRKGA, *Biased Random-Key Genetic Algorithm*) é um RKGA, com algumas particularidades. Assim como o RKGA, as meta-heurísticas BRKGA obtém uma solução viável para um problema de otimização através da decodificação de uma solução codificada, como um vetor de chaves aleatórias geradas no intervalo

real [0, 1). Após o cálculo da função objetivo realizado pelo decodificador e o particionamento dos indivíduos em dois grupos (elite e não-elite), cria-se uma nova geração. De modo semelhante, uma estratégia elitista garante que as melhores soluções obtidas sejam passadas sem alterações de uma geração para a próxima, enquanto a mutação de indivíduos desempenha o papel de perturbar a convergência da população esquivando de valores ótimos locais.

Diferentemente do RKGA, o BRKGA seleciona os indivíduos para a operação de cruzamento de modo tendencioso. Neste método, determina-se que o cruzamento de indivíduos (pais) no algoritmo seja necessariamente feito com um indivíduo de elite e um indivíduo não-elite, ao invés de escolher ambos a partir da população toda. A repetição na seleção de um indivíduo para cruzamento é permitida e, portanto, um pai pode gerar mais de um descendente na mesma geração. Ainda, admite-se que a probabilidade do indivíduo descendente herdar a chave do pai elite é sempre maior em relação ao pai não-elite.

Diversos trabalhos propõem abordagens BRGKA para diferentes tipos de problemas de otimização. Em Ericsson, Resende e Pardalos (2002) foram propostos algoritmos genéticos para resolver o problema de configuração de peso do protocolo de roteamento de internet intra-domínio. O método BRKGA dos autores, denominado por GAOSPF (*Genetic Algorithm for Open Shortest Path First routing*) trata o problema de roteirização em redes com o objetivo de minimizar o custo total na distribuição do fluxo de dados.

Um BRKGA foi comparado com algoritmos genéticos da literatura em Gonçalves e Resende (2004). A meta-heurística para projetar células de manufatura, apresentada pelos autores, obteve os melhores resultados na comparação realizada. No ano seguinte, em Gonçalves, Mendes e Resende (2005), foi proposto pelos autores um BRKGA para o problema de programação de tarefas no ambiente *job shop* com o objetivo de minimizar o *makespan*, isto é, obter o menor tempo de processamento de todas as tarefas. Em Gonçalves (2007) foi proposto um algoritmo genético para o problema de empacotamento bidimensional com objetivo de reduzir a área não ocupada na solução geométrica. O algoritmo proposto pelo autor é uma abordagem híbrida composto por um procedimento de colocação e um BRKGA.

Para resolver o problema de congestionamento em redes rodoviárias, um BRKGA foi proposto em Buriol et al. (2010) com o objetivo de otimizar o desempenho da rede de transportes através da alocação estratégica de pedágios. O BRKGA também foi utilizado em Valente et al. (2011), porém em outro contexto. Os autores propuseram um BRKGA para o problema de programação de tarefas no ambiente de uma máquina com objetivo de minimizar atrasos. Já em Noronha, Resende e Ribeiro (2011), utilizou-se o BRKGA para o problema de roteirização e atribuição de comprimentos de onda, minimizando a quantidade total de comprimentos.

Um tutorial para a implementação do BRKGA foi estruturado em Gonçalves e Resende (2011). No ano seguinte, um algoritmo para o empacotamento tridimensional, denominado por BRKGA-CLP (*single Container Loading Problem*), foi proposto pelos autores em Resende et al. (2012), com o objetivo de otimizar o volume de contêineres. Em Mainieri (2014) foi utilizado um

BRKGA em um ambiente de trabalho sequencial em estágio, conhecido pelo termo *Flowshop híbrido*, em que cada estágio deve ser executado apenas uma vez e por uma máquina. O método foi proposto para minimizar o atraso de todos os trabalhos realizados no ambiente dado. Também com o objetivo de minimização, porém em uma aplicação completamente distinta, em Stefanello (2015) um BRKGA foi proposto para resolver o problema de tráfego em redes de transporte e de telecomunicações com o objetivo de controlar o fluxo na rede, minimizando o congestionamento.

Prasetyo et al. (2016) realizaram uma revisão o desenvolvimento de BRKGAs para resolver diversos problemas de otimização. O trabalho teve como objetivo fornecer uma contextualização do desenvolvimento dos algoritmos e áreas de aplicação encontradas na literatura. Os autores destacaram a tendência de aumento do uso do BRKGA para melhorar o desempenho de outros procedimentos quando combinados. Em Mönch e Roob (2018) foi proposta uma estrutura matheurística para problemas de escalonamento com máquinas de processamento em lote paralelo, com atribuição de lotes às máquinas e sequenciamento realizados por uma abordagem BRKGA. Apesar das limitações indicadas no trabalho, a estrutura proposta foi capaz de fornecer soluções de alta qualidade em um curto período de tempo computacional.

Em Mauri et al. (2021) um BRKGA foi proposto, juntamente com outra meta-heurística híbrida, para resolver um problema de localização de instalações de multiprodutos, com o objetivo de minimizar o custo relacionado a fábricas e depósitos abertos mais o custo de transporte dos produtos das fábricas até os clientes. O BRKGA também foi considerado em publicações ainda mais recentes, como em Lima et al. (2021) e Guimarães et al. (2022).

Em todos os trabalhos supracitados, o uso do BRKGA gerou bons resultados, obtendo melhor desempenho frente a outros algoritmos genéticos encontrados na literatura, indicando o alto potencial do método. Além de sua qualidade e facilidade de adaptação para abranger diversos problemas, o BRKGA tem garantia de viabilidade de soluções (GONÇALVES; RESENDE, 2011) e possui tutoriais disponíveis que facilitam sua implementação. Deste modo, um BRKGA para o problema de planejamento de circuitos VLSI é proposto e detalhado no Capítulo 5 deste trabalho.

3.2 Circuitos integrados VLSI

Zhan, Feng e Sapatnekar (2006) apresentaram um algoritmo analítico sofisticado de planejamento de *layout* de circuitos VLSI que pode ser usado para empacotar eficientemente módulos flexíveis em uma matriz fixa. A localização e o dimensionamento dos módulos são otimizados para que seja alcançado valor reduzido de comprimento total de fio. Primeiramente, as posições de cada módulo são determinadas de forma aproximada. Em seguida, as sobreposições são gradualmente removidas no segundo estágio para obter a alocação final. Apesar da abordagem lidar bem com projetos de grande escala, ela não pode garantir uma solução sem sobreposição.

Young et al. (2001) utilizaram relaxação lagrangeana para resolver o problema de alocação de módulos retangulares flexíveis. O método desenvolvido teve por objetivo minimizar a área

total de empacotamento dos módulos, e foi implementado em um *simulated annealing framework* utilizando a representação por pares de sequência. Com uma abordagem semelhante, Chi e Chi (2002) propuseram um algoritmo *floorplanning* com módulos flexíveis para otimizar a área utilizada no design de um circuito VLSI. Chen et al. (2007) estudaram o problema no *floorplannig* com contorno fixo. Um algoritmo baseado em uma busca evolutiva robusta foi desenvolvido, juntamente com uma nova abordagem para flexibilizar os módulos alocados e, assim, ajustar o *layout* reduzindo o contorno fixo no chip.

Chen e Chang (2009) propuseram uma abordagem analítica exata tratando as posições relativas dos módulos retangulares como variáveis binárias do problema no *floorplanning*. Para minimizar a área de forma linear, a largura do retângulo envolvente foi fixada e a altura necessária para a alocação dos módulos minimizada na função objetivo, respeitando limites pré-definidos. O modelo matemático formalizado pelos autores apresentou resultados satisfatórios, podendo considerar módulos flexíveis e gerar soluções exatas. Apesar disso, a formulação é um problema linear inteiro misto e seu tempo de execução se torna bastante elevado para médias e grandes instâncias. Mais detalhes sobre esta abordagem é apresentada na Seção 3.3, dado que a abordagem dos autores Chen e Chang (2009) foi uma forte referência para o desenvolvimento do modelo proposto na Capítulo 4 deste trabalho.

Chen, Zhu e Ali (2011) sugeriram um algoritmo híbrido de *simulated annealing* (HSA) para o planejamento de alocação na etapa física de circuitos VLSI. O HSA usa um método guloso para construir uma árvore B* inicial, uma operação em árvore B* para explorar o espaço de busca e uma estratégia de busca para aprimorar os resultados da exploração feita. Para reduzir a medida de área de empacotamento, além do comprimento de fio na alocação de módulos rígidos, Anand, Saravanasankar e Subbaraj (2013) implementaram uma técnica de otimização multiobjetivo chamada AMOSA *(Archived Multiobjective Simulated Annealing Algorithm)*, um algoritmo baseado em *simulated annealing*.

Chan, Hsu e Lin (2013) adotaram uma abordagem de dois estágios para o *floorplanning* de um conjunto de módulos de tamanhos variados. O primeiro estágio consiste na distribuição dos módulos em um contorno fixo otimizando o comprimento de fio utilizado. No segundo estágio, é realizada uma abordagem baseada em partição para obter uma solução viável sem prejudicar o bom resultado atingido no estágio inicial. He, Ji e Li (2015) apresentaram uma heurística de redução dinâmica para o problema de minimização da área de empacotamento de retângulos flexíveis em circuitos. A abordagem proposta transforma a instância do problema original em uma série de instâncias menores, determinando dinamicamente as dimensões do retângulo envolvente. Com o mesmo objetivo de estudo, Gracia e Rajaram (2015) adotaram uma estratégia de busca híbrida e algoritmo genético para otimizar o *floorplanning* VLSI.

Laskar et al. (2015) apresentaram um estudo e comparação dos diferentes algoritmos de otimização e as representações envolvidas no problema de *layout* do VLSI. Dentre os algoritmos, a maior parte do trabalho foi realizada para otimização de minimização de área. Singh, Baghel

e Agarwal (2016) reuniram e revisaram heurísticas e algoritmos meta-heurísticos de trabalhos que consideram o problema de otimização no *floorplanning* de VLSI. Comparações entre os procedimentos reunidos foram incluídas neste estudo e mesmo a representação mais eficaz apresentada indicou limitações de flexibilidade.

Wu et al. (2016) propuseram um procedimento heurístico para resolver o problema de minimização da área de empacotamento de retângulos quando o *layout* final deveria conter um retângulo central, com flexibilidade nas dimensões, as quais poderiam ser alteradas dentro de limites razoáveis. Estratégias de preenchimento do espaço interno foram propostas para melhorar a taxa de preenchimento do *layout* final. Jenifer, Anand e Levingstan (2016) propuseram uma abordagem para a determinação do design físico do VLSI. Com o objetivo de obter a área mínima do *floorplanning* remodelando os blocos presentes, o trabalho proposto redefine o problema com uma meta-heurística baseada em *simulated annealing*.

Singh, Baghel e Agarwal (2016) usaram um algoritmo de *simulated anneling* otimizar a área do problema de planejamento do *floorplanning* de VLSI, reduzindo o espaço interno não utilizado na alocação dos módulos. Para resolver o problema de empacotamento de retângulos flexíveis, Ji et al. (2017) propuseram um algoritmo que funde todos os retângulos em um retângulo final composto. Os autores mostraram que o algoritmo proposto pode garantir um *layout* viável sob algumas condições.

Com o objetivo final de reduzir a área total de alocação de módulos no planejamento de *layouts* de circuitos VLSI de contorno fixo, Venkatraman e Sundhararajan (2019) propuseram uma abordagem híbrida que combina um algoritmo genético com o algoritmo *Harmony Search* (HS). Laudis e Ramadass (2020) apresentaram um algoritmo multiobjetivo bioinspirado que explora um conjunto de soluções chamado de geração. A cada geração parte da população sofre mutações e cruzamentos, procurando atingir uma nova solução melhor que as anteriores. Shunmugathammal, Columbus e Anand (2020) propuseram um método para resolver o problema de otimização de *floorplanning* usando algoritmo genético que é chamado *Lion Optimization Algorithm* (LOA) e tem como objetivo minimizar a área utilizada na alocação. O LOA foi desenvolvido para *layouts* não-guilhotinados de itens com restrição de contorno fixo.

Apesar dos diversos trabalhos abordando o problema de empacotamento de retângulos, métodos de solução mais eficientes podem ser explorados, garantindo que muitos estudos ainda sejam desenvolvidos e possibilitando o uso de instâncias mais próximas do ponto de vista prático, principalmente no contexto do posicionamento de módulos flexíveis em circuitos VLSI. Para resolver este problema, abordagens exata, matheurística e meta-heurística baseadas na literatura serão estudadas neste trabalho.

3.3 Modelo de Chen e Chang (2009)

Para solucionar o problema de empacotamento de módulos retangulares, fixos ou flexíveis, Chen e Chang (2009) propuseram uma abordagem analítica exata tratando as posições relativas de itens retangulares como variáveis binárias do problema. Sua formulação inclui uma função objetivo e considera dois conjuntos de restrições básicas: restrições de não sobreposição do item e restrições de dimensão. Assim, garante-se que os módulos (itens retangulares) alocados não estejam sobrepostos entre si e estejam completamente contidos no retângulo envolvente.

Com duas variáveis binárias (p_{ij} e q_{ij}) introduzidas, dois módulos i e j são considerados não sobrepostos, se pelo menos um dos casos apresentados na Tabela 1 for satisfeito. Na formulação, (x_i, y_i) é o par ordenado que indica a posição do canto inferior esquerdo do item i, w_i a largura e h_i a altura do item i. Analogamente, para o item j.

Tabela 1 – Casos possíveis de posicionamento de dois módulos i e j, com $i < j$.

		p_{ij}	q_{ij}
i está à esquerda de j	$x_i + w_i \leq x_j$	0	0
i está abaixo de j	$y_i + h_i \leq y_j$	0	1
i está à direita de j	$x_i - w_i \geq x_j$	1	0
i está acima de j	$y_i - h_i \geq y_j$	1	1

Fonte: Traduzido de Chen e Chang (2009).

O modelo matemático apresentado por Chen e Chang (2009), tem como objetivo minimizar a função não linear definida pela área do *layout*, $A = x \cdot y$, em que x é a largura e y a altura do retângulo envolvente resultante após a alocação dos itens. O valor de x e y deve ser obtido, respeitando limites superiores pré-definidos de largura e altura do retângulo envolvente do *layout*, respectivamente, W e H. Para transformar a função objetivo em linear, diminuindo a complexidade do sistema e possibilitando tratar o problema como linear inteiro misto, uma aproximação para o problema foi feita. Assim, fixou-se a largura $x = W$, minimizando apenas y, e consequentemente, reduzindo a área total utilizada na alocação. Tal modelo é definido por:

Parâmetros:

- n: número de itens;
- $J = \{1, \ldots, n\}$: conjunto de itens i;
- w_i, h_i: dimensões do item i;
- W, H: dimensões máximas do retângulo envolvente (largura e altura do chip).

Variáveis:

- x_i, y_i: coordenadas do item i (variável de posição);
- p_{ij}, q_{ij}: relação horizontal e vertical de posição, para todo $i < j$.

Modelo:

$$\begin{aligned}
&minimizar && y && && (3.1)\\
&sujeito\ a: && && && \\
& && x_i + w_i \leq W && \forall i \in J && (3.2)\\
& && y_i + h_i \leq y && \forall i \in J && (3.3)\\
& && x_i + w_i \leq x_j + W(p_{ij} + q_{ij}) && i < j,\ \forall i \in J && (3.4)\\
& && y_i + h_i \leq y_j + H(1 + p_{ij} - q_{ij}) && i < j,\ \forall i \in J && (3.5)\\
& && x_i - w_j \geq x_j - W(1 - p_{ij} + q_{ij}) && i < j,\ \forall i \in J && (3.6)\\
& && y_i - h_j \geq y_j - H(2 - p_{ij} - q_{ij}) && i < j,\ \forall i \in J && (3.7)\\
& && x_i, y_i \geq 0 && \forall i \in J && (3.8)\\
& && p_{ij}, q_{ij} \in \{0, 1\} && i < j,\ \forall i \in J && (3.9)
\end{aligned}$$

No modelo (3.1)-(3.9), a função objetivo (3.1) minimiza a altura do retângulo envolvente do *layout*. As restrições (3.2) e (3.3) garantem que as dimensões dos itens alocados estejam dentro dos limites de largura e altura, respectivamente. As restrições (3.4)-(3.7) asseguram que os módulos, tomados dois a dois, não fiquem sobrepostos e que uma das desigualdades exibidas na Tabela 1 seja aplicada. Quando $p_{ij} = 1$ e $q_{ij} = 0$, por exemplo, a restrição (3.6) é dada por $x_i - w_j \geq x_j - W(1 - 1 + 0)$ e a inequação $x_i - w_j \geq x_j$ é aplicada, no caso em que i está à direita de j. A restrição (3.8) garante a não-negatividade das variáveis x_i, y_i e a restrição (3.9) define o domínio das variáveis p_{ij} e q_{ij}.

O modelo (3.1)-(3.9) não envolve objetivo ou restrições específicas do *floorplanning*. Desta forma, pode ser utilizado genericamente para problemas de empacotamento que buscam minimizar altura ou largura do retângulo envolvente. Com base na formulação de Chen e Chang (2009), que considera apenas a otimização da área, adaptações podem ser realizadas com a finalidade de melhor representar o problema de planejamento de *layouts* em circuitos integrados VLSI. Um modelo matemático é proposto e detalhado no Capítulo 4.

4 Modelo Matemático

Neste capítulo, conceitos essenciais e um modelo matemático são propostos para resolver o *floorplanning* em circuitos VLSI, abordado como um problema de empacotamento de retângulos flexíveis. Este problema considera como variáveis de decisão as posições ideais para um determinado conjunto de módulos retangulares flexíveis alocados em uma área de chip, de modo que não haja sobreposição, além de decidir quais as dimensões mais adequadas para cada retângulo, respeitando proporções pré-estabelecidas (*aspect radio*) entre sua altura e largura.

4.1 Contextualização

Com o avanço da tecnologia, o número de módulos em um chip aumenta. Com o aumento da complexidade do *floorplanning* e dos requisitos do circuito, novas considerações podem ser agregadas ao problema, entre elas a possibilidade dos módulos alocados serem flexíveis. Para considerar esta possibilidade, no modelo proposto por Chen e Chang (2009) deve-se incluir as restrições (4.1) e (4.2):

$$w_i^{min} \leq w_i \leq w_i^{max} \quad \forall i \in J \tag{4.1}$$

$$h_i^{min} \leq h_i \leq h_i^{max} \quad \forall i \in J \tag{4.2}$$

Os módulos flexíveis podem alterar suas alturas e larguras, ao contrário dos módulos rígidos com alturas e larguras fixas, mantendo a mesma área do módulo. Neste caso, as dimensões w_i e h_i passam a ser tratadas como variáveis no modelo e novos parâmetros são considerados: w_i^{min} e w_i^{max} indicam, respectivamente, o limite mínimo e máximo da largura do item i. Analogamente, h_i^{min} e h_i^{max} indicam, respectivamente, o limite mínimo e máximo da altura do item i. Com módulos ajustáveis, configurações de *layout* mais eficientes são obtidas, possibilitando um melhor uso da área do chip e encaixe na alocação dos itens.

Além da área utilizada na alocação, o comprimento de fio utilizado para conectar os módulos no circuito também deve ser considerado. Uma medida de comprimento total de interconexões muito grande resulta em chips maiores e mais lentos, afetando o desempenho e o custo de fabricação do circuito. Ao reduzir a quantidade de fio utilizado, obtém-se circuitos que

utilizam menos recursos de roteamento, com tempos de rota mais consistentes e com maior nível de desempenho (KAHNG et al., 2011; CHAN; HSU; LIN, 2013).

4.2 Representação de posicionamento

Para representar a configuração geométrica dos módulos no *layout*, as posições dos itens podem ser especificadas por coordenadas definidas em relação a um sistema de eixos ou através de um conjunto de relações topológicas entres os itens, definindo as posições relativas entre eles. Neste trabalho, foi utilizada uma representação derivada da ferramenta padrão, conhecida como par de sequência (MURATA et al., 1996), para especificar a posição relativa entre cada par de módulos e obter um empacotamento sem sobreposição. As posições relativas atribuídas aos itens podem ser escritas como inequações e facilmente inseridas em um modelo matemático.

Assim como em Chen e Chang (2009), dado dois módulos i e j, as variáveis binárias de posicionamento l_{ij} e b_{ij}, horizontal e vertical, respectivamente, são introduzidas para denotar a posição que o item i ocupa em relação ao item j. Apenas uma das variáveis binárias associadas ao par de itens assume valor não nulo, assim, os módulos i e j são considerados não sobrepostos se uma das condições apresentadas na Tabela 2 é válida. Possíveis casos de posicionamento de itens são ilustrados na Figura 5.

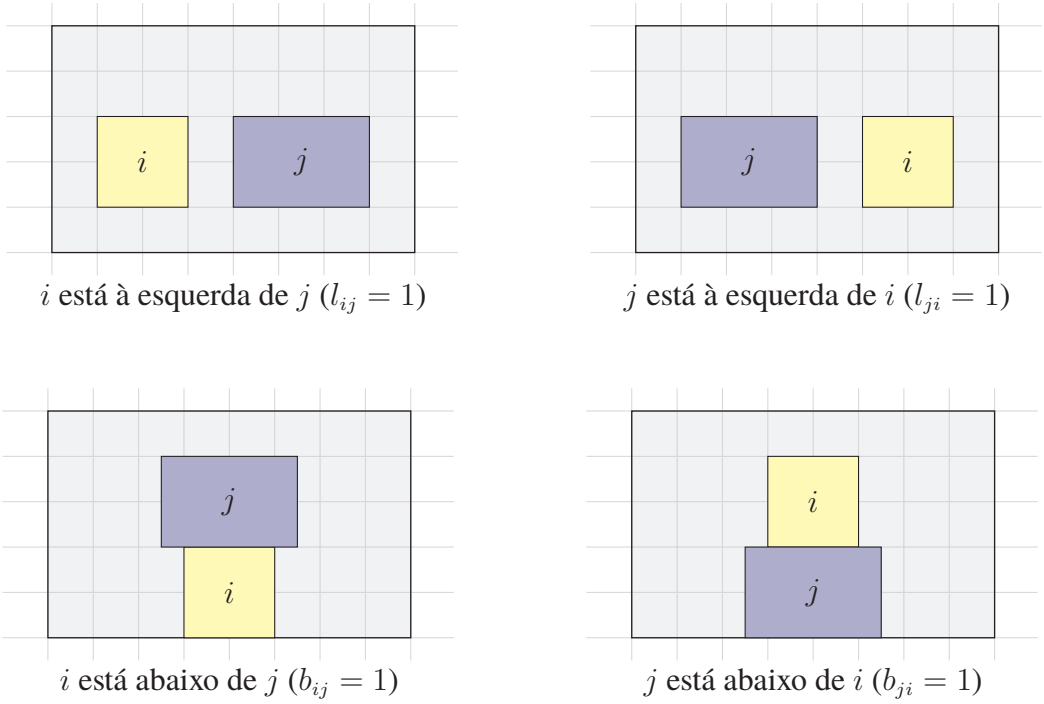


Figura 5 – Casos possíveis de posicionamento entre dois módulos i e j.

Dado que a ferramenta de Murata et al. (1996) é a mais utilizada para representar padrões de posicionamento ao longo da literatura, a representação de posicionamento descrita acima foi adotada neste trabalho por se relacionar fortemente com pares de sequência. Desta forma, é

Tabela 2 – Valores das variáveis binárias ao posicionar dois módulos i e j, com $i < j$

	l_{ij}	b_{ij}	l_{ji}	b_{ji}
i está acima de j	0	0	0	1
i está à direita de j	0	0	1	0
i está abaixo de j	0	1	0	0
i está à esquerda de j	1	0	0	0

possível adaptar ou incrementar os métodos aqui propostos, combinando-os com outros trabalhos existentes sem grandes complicações na representação de posicionamento dos itens, também chamados de módulos.

4.3 Cálculo do comprimento de fio

Nos circuitos integrados, uma rede r é um conjunto de módulos (itens retangulares flexíveis) interconectados pré-definidos na instância. Para calcular o comprimento do fio gasto para fazer essas conexões, recorre-se à aproximação, dado que as informações reais de roteamento não estão disponíveis no estágio do *floorplanning*. Uma aproximação de comprimento de fio comumente usada para uma rede é medir seu comprimento de fio de meio-perímetro ponderado (HPWL). O método HPWL é razoavelmente preciso, calculado com eficiência e adota como distância o comprimento de meio perímetro da menor caixa delimitadora (retângulo) que inclui todos os pontos de conexão do circuito integrado, também chamados de pinos. Quando as posições dos pinos não são fornecidas, pode-se calcular o HPWL usando os centros dos módulos flexíveis (CHEN; CHANG, 2009; KAHNG et al., 2011).

Neste trabalho, o cálculo do HPWL é obtido a partir do comprimento de meio perímetro do retângulo exterior com arestas que contém o centro dos módulos que estão ligados entre si, isto é, a distância entre os dois pontos centrais dos módulos é dada pela soma das diferenças absolutas de suas coordenadas, utilizando apenas retas verticais e horizontais. Este formato não-euclidiano de obter distâncias d_x e d_y é conhecido como Geometria do Táxi ou Distância de *Manhattan* (KAHNG et al., 2011; JENIFER; ANAND; LEVINGSTAN, 2016), ilustrado na Figura 6. O comprimento de fio entre dois itens $i, j \in J$ é dado por $d = d_x + d_y$, sendo:

$$d_x = |x_j - x_i| \tag{4.3}$$

$$d_y = |y_j - y_i| \tag{4.4}$$

Para obter as dimensões W_r e H_r, respectivamente, largura e altura de uma rede r, calcula-se a soma da distância central de todos os módulos interconectados. O modelo matemático proposto otimiza as estimativas dessas métricas de comprimento de fio, minimizando as dimensões das redes do circuito integrado VLSI.

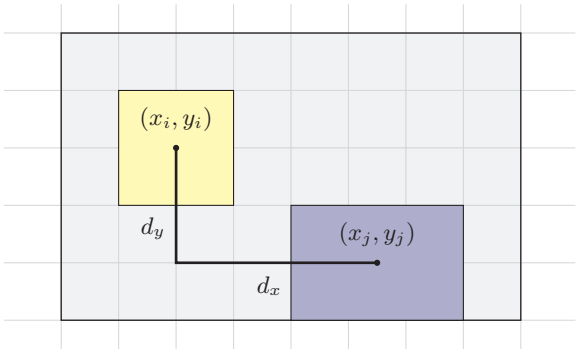


Figura 6 – Comprimento de fio HPWL entre dois itens $(d_x + d_y)$.

Nas Figuras 7 e 8, um conjunto de itens alocados são ilustrados para exemplificar como a distância de HPWL é, geometricamente, obtida em uma rede. Cada item possui seu ponto central destacado e duas redes distintas são apresentadas. Na Figura 7, o retângulo envolvente que contém os itens 1, 2 e 4 representa uma possível rede (Rede 1) e espera-se que todos os itens contidos nela sejam interconectados. As dimensões $W_1 = 7$ e $H_1 = 4$, largura e altura da Rede 1, respectivamente, também são apresentadas.

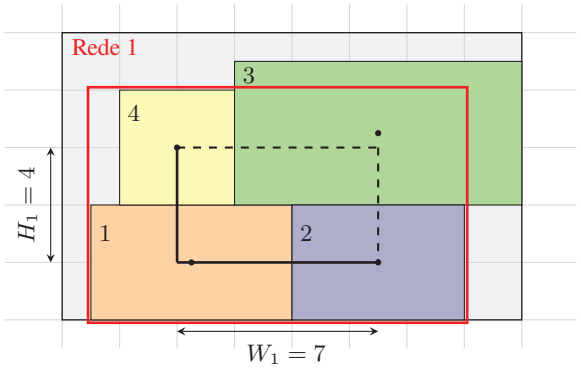


Figura 7 – Exemplo 1 de distância HPWL = 11.

O comprimento de fio, ou distância HPWL, da rede é dado pela soma de suas dimensões, nesse caso, o comprimento de fio da Rede 1 é igual a 11 unidades de medida. Na Figura 8, a rede apresentada (Rede 2) contém os itens 3 e 4, conectados entre si com comprimento de fio em destaque. As métricas associadas à Rede 2 são obtidas de forma análoga à Rede 1, com comprimento de fio igual à 7,5 unidades de medida.

Sabendo que o comprimento de fio, utilizado nas interconexões, tem grande influência nos custos e desempenho de um circuito, busca-se minimizar as distâncias associadas ao problema de empacotamento de retângulos flexíveis abordado. Ao adotar uma função objetivo que reduz a distância HPWL do circuito VLSI, espera-se obter *layouts* compactos e eficientes.

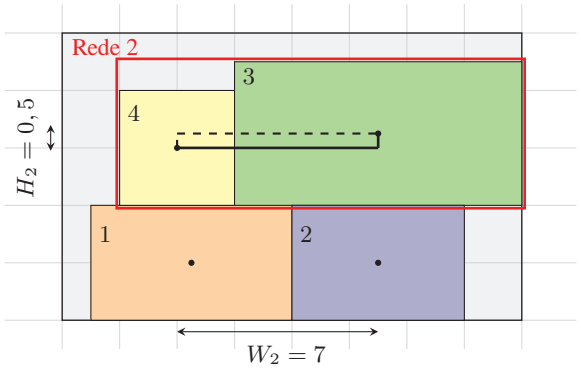


Figura 8 – Exemplo 2 de distância HPWL = 7,5.

4.4 Modelo Matemático

A formulação proposta tem como objetivo minimizar as distâncias do comprimento de fio HPWL entre subconjuntos especificados dos itens, isto é, reduzir a medida da quantidade de metal necessária para interconectar os módulos de forma a garantir a funcionalidade do chip. Os parâmetros e variáveis do modelo matemático são definidos por:

Parâmetros:

- n: número de itens flexíveis (módulos);
- m: número de redes;
- $J = \{1, \ldots, n\}$: conjunto de itens i;
- $R = \{1, \ldots, m\} \subseteq J$: conjunto de redes r;
- w_i^{min}, h_i^{min}: dimensões mínimas do item i;
- w_i^{max}, h_i^{max}: dimensões máximas do item i;
- a_i^{min}, a_i^{max}: proporção mínima e máxima entre altura e largura do item i (*aspect ratio*);
- W^{max}, H^{max}: dimensões máximas do retângulo envolvente (largura e altura do chip).

Variáveis:

- x_i: abcissa do item i (variável de posição);
- y_i: ordenada do item i (variável de posição);
- w_i: largura do item i;

- h_i: altura do item i;
- W_r: largura da rede r;
- H_r: altura da rede r;
- l_{ij} e b_{ij}: relação horizontal (*left*) e vertical (*below*) de posição, respectivamente;

$$l_{ij} = \begin{cases} 1, \text{ se } i \text{ está à esquerda de } j; \\ 0, \text{ caso contrário.} \end{cases} \qquad b_{ij} = \begin{cases} 1, \text{ se } i \text{ está abaixo de } j; \\ 0, \text{ caso contrário.} \end{cases}$$

Modelo:

$$minimize$$

$$\sum_{r \in R} W_r + H_r \tag{4.5}$$

$$sujeito\ a:$$

$$w_i^{min} \leq w_i \leq w_i^{max} \qquad \forall i \in J \tag{4.6}$$

$$h_i^{min} \leq h_i \leq h_i^{max} \qquad \forall i \in J \tag{4.7}$$

$$a_i^{min} \cdot h_i \leq w_i \leq a_i^{max} \cdot h_i \qquad \forall i \in J \tag{4.8}$$

$$x_j \geq x_i + w_i - (1 - l_{ij}) \cdot W^{max} \qquad \forall i \neq j \in J \tag{4.9}$$

$$y_j \geq y_i + h_i - (1 - b_{ij}) \cdot H^{max} \qquad \forall i \neq j \in J \tag{4.10}$$

$$l_{ij} + b_{ij} + l_{ji} + b_{ji} = 1 \qquad \forall i < j \in J \tag{4.11}$$

$$W_r \geq |(x_i + \frac{w_i}{2}) - (x_j + \frac{w_j}{2})| \qquad \forall r \subseteq R, \forall i, j \in r \tag{4.12}$$

$$H_r \geq |(y_i + \frac{h_i}{2}) - (y_j + \frac{h_j}{2})| \qquad \forall r \subseteq R, \forall i, j \in r \tag{4.13}$$

$$l_{ij}, b_{ij} \in \{0, 1\} \qquad \forall i \neq j \in J \tag{4.14}$$

No modelo (4.5)-(4.14), a função objetivo (4.5) minimiza o comprimento de fio utilizado nas interconexões dos itens, de todas redes, adotando a simplificação de distância entre dois módulos que, neste caso, é sempre calculada entre os centros geométricos dos módulos. As restrições (4.6) e (4.7) garantem que as dimensões dos módulos flexíveis estejam dentro do intervalo permitido. Com dimensões variáveis, é preciso que a largura (w_i) de cada item i seja limitada inferior e superiormente, respeitando os parâmetros pré-estabelecidos. Analogamente, a altura (h_i) possui valor mínimo e máximo. As restrições (4.8) estabelecem que cada módulo retangular deve respeitar uma proporção altura-largura ($a_i = h_i/w_i$) para manter seu funcionamento e eficiência ao ser flexibilizado.

As restrições (4.9) e (4.10), referentes às relações de posicionamento entre os itens, garantem que não haja sobreposição entre dois módulos e que ambos estejam contidos no retângulo envolvente do *layout*, isto é, estejam inteiramente alocados na região do chip. No caso

de $l_{ij} = 1$, temos que $x_j \geq x_i + w_i$, portanto, a abcissa de j deve assumir um valor igual ou superior a abcissa do ponto inferior esquerdo do item i, sem que haja sobreposição horizontal dos itens. Caso $l_{ij} = 0$, $x_j \geq x_i + w_i - W_{max}$ garante que o item j está horizontalmente contido no retângulo envolvente. De modo análogo, as restrições (4.10) tratam as posições dos itens verticalmente.

As restrições (4.11) asseguram que uma, e apenas uma, das variáveis binárias de posicionamento referentes a i e j assuma valor não nulo. Desta forma, é possível que o item i esteja à esquerda e abaixo de j simultaneamente, porém este caso é indicado somente através da transitividade entre ambos os itens e os demais. Por exemplo, dados os itens i, j e k, é possível inferir que, se i está abaixo de k ($b_{ik} = 1$) e k está abaixo de j ($b_{kj} = 1$), certamente i também está abaixo de j, ainda que $b_{ij} = 0$ e $l_{ij} = 1$. Assim, i e j aparecem relacionados verticalmente por transitividade e horizontalmente através de uma das variáveis binárias de posicionamento que assume valor não nulo.

As restrições (4.12) e (4.13) são referentes às dimensões de rede, calculadas, rede a rede, para todos os itens contidos nela. A largura da rede assume o valor da maior diferença das abcissas dos pontos centrais dos módulos pertencentes à ela. Da mesma forma, a altura da rede é obtida em função das ordenadas dos pontos centrais de cada par de itens. Cada um desses grupos de restrições obtém as dimensões do semi-perímetro do retângulo que contém os pontos centrais dos módulos na rede. As restrições (4.14) definem l_{ij} e b_{ij} o domínio das variáveis de decisão.

O modelo proposto foi baseado em Chen e Chang (2009). As formulações diferem na função objetivo, além das restrições de dimensão de rede (4.12) e (4.13) que foram acrescentadas para considerar a minimização do comprimento de fios ao invés de otimizar a área do retângulo envolvente. Adequações também foram feitas nas restrições (4.9), (4.10) e (4.11) de posicionamento entre os módulos para melhor atender a finalidade do problema considerado neste trabalho, dado que as variáveis binárias assumem características distintas na formulação proposta.

5 Abordagens heurísticas

O problema de empacotamento de retângulos flexíveis pode ser facilmente representado através de modelos matemáticos, porém é difícil de ser solucionado devido ao grande número de variáveis envolvidas. Obter soluções exatas implica em alta complexidade computacional, principalmente para problemas com grandes dimensões. Assim, métodos heurísticos são frequentemente utilizados para explorar o espaço de soluções com um baixo esforço computacional e garantia de boas soluções (SINGH; BAGHEL; AGARWAL, 2016; WU et al., 2016).

Neste capítulo, uma abordagem heurística de janela deslizante (*SW, Sliding Window*) e uma mateurística BRKGA são propostas para resolver de maneira eficiente o problema de empacotamento de retângulos flexíveis no contexto do *floorplanning* de circuitos VLSI. Usada em ambos os métodos heurísticos, a estratégia de janela deslizante consiste em pegar, sob condições específicas, um subconjunto de uma lista dada. A cada iteração um elemento é substituído no subconjunto, criando um efeito de deslizamento da janela sobre o conjunto.

5.1 Heurística SW

Combinando programação matemática e um método heurístico para reduzir a dificuldade de resolução do problema, a heurística proposta é um procedimento iterativo que conta com uma estratégia de inicialização, estabelecida utilizando critérios de ordenação para os itens que se deseja alocar. Desta forma, é possível definir quais módulos serão priorizados no planejamento do circuito e resolver um problema parcial, com apenas alguns dos módulos a serem alocados, ao invés de buscar uma solução exata para o problema. A cada iteração, um novo item não-alocado é selecionado para ser inserido no *layout* e um sub-modelo (4.5)-(4.14) é criado com tamanho fixo e reduzido em relação ao modelo do problema original.

Para formalizar a heurística SW, novos parâmetros foram adotados:

- sw: tamanho da janela deslizante, $sw \in \{1, \cdots, n-1\}$;
- con_i: conectividade do item i, $con_i \in \{0, \cdots, m\}$;
- it: índice de iteração, $it \in \{0, \cdots, n-sw\}$.

O tamanho da janela deslizante sw indica a quantidade de itens com posições livres, e suas variáveis de decisão correspondentes, considerados na resolução do sub-modelo, isto é, o número de módulos que podem ser posicionados, reposicionados ou ajustados em suas dimensões a cada iteração. O parâmetro sw pode assumir valor entre 1 e $n-1$, sendo n a quantidade total de itens do problema. A conectividade con_i de um item i indica o número de redes r que o contém, calculado no momento de inicialização do algoritmo e limitado no intervalo de 0 a m, sendo m o número total de redes do problema. Os valores sw e con_i são constantes durante todo o processamento da heurística. Na primeira etapa da heurística SW, é utilizada uma estratégia de inicialização que consiste em ordenar os módulos flexíveis seguindo um critério de ordenação, dentre os critérios:

- área de forma não-decrescente;
- área de forma não-crescente;
- conectividade de forma não-decrescente;
- conectividade de forma não-crescente.

Na etapa seguinte, um sub-modelo é criado para resolver a alocação de parte dos itens do problema original na primeira iteração $(it = 0)$. Este sub-modelo é criado utilizando (4.5)-(4.14). Apenas os sw primeiros itens da lista ordenada são considerados na resolução, diminuindo significativamente a dificuldade computacional do problema. O sub-modelo é resolvido e, a partir da sub-solução obtida, fixam-se as posições relativas do primeiro item da alocação feita, isto é, as variáveis l e b associadas ao item passam a ser parâmetros no sub-modelo. Após acrescentar o próximo item não-alocado da lista ordenada ao sub-modelo, inicia-se uma nova iteração $(it = 1)$. O sub-modelo é resolvido e, a partir da sub-solução obtida, fixam-se as posições relativas l e b do segundo item da lista ordenada.

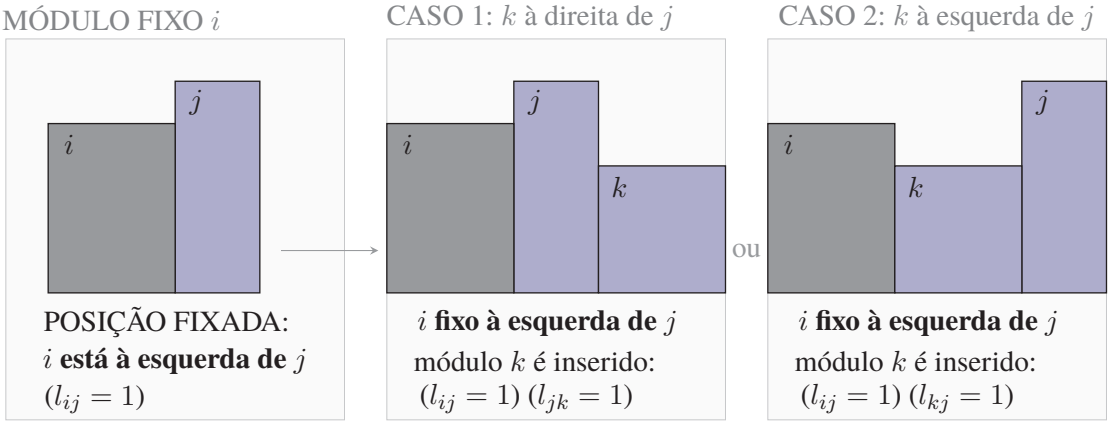


Figura 9 – Representação horizontal de posicionamento relativo fixado do item i (módulo fixo) com inserção do item k (módulo livre) à direita ou à esquerda de j (módulo livre).

Para facilitar a notação, assume-se que um módulo livre é um item que tem suas posições relativas livres (l_{ij}, l_{ji}, b_{ij} e b_{ji}), assim, pode ser posicionado de forma independente no *layout*. Por outro lado, chama-se de módulo fixo o item que tem suas posições relativas fixadas, isto é, caso o módulo fixo i esteja à esquerda de um módulo j, é necessário que essa relação seja mantida em todas as iterações futuras. Na fixação das posições relativas associadas ao item i com $l_{ij} = 1$, por exemplo, ainda que um novo item k seja acrescentado ao *layout* à direita de i, a relação entre i e j deve ser mantida. Observe o esquema na Figura 9 com a representação da relação horizontal entre os módulos i, j e k, com $l_{ij} = 1$ fixo e variação entre $l_{kj} = 1$ e $l_{jk} = 1$.

O sub-modelo é atualizado, acrescentando o terceiro item não-alocado, e resolvido. A partir da sub-solução obtida, fixam-se as posições relativas referentes ao item $it + 1$, sendo it o índice da iteração presente, e atualiza-se o sub-modelo inserindo o próximo item não-alocado da lista. O sub-modelo é resolvido novamente. Estes passos são realizados de modo iterativo até que a quantidade de itens do sub-modelo seja igual à quantidade de itens do modelo inicial. Adota-se como solução do modelo a solução obtida para o sub-modelo na última iteração realizada. A Figura 10 apresenta brevemente as etapas da abordagem heurística SW.

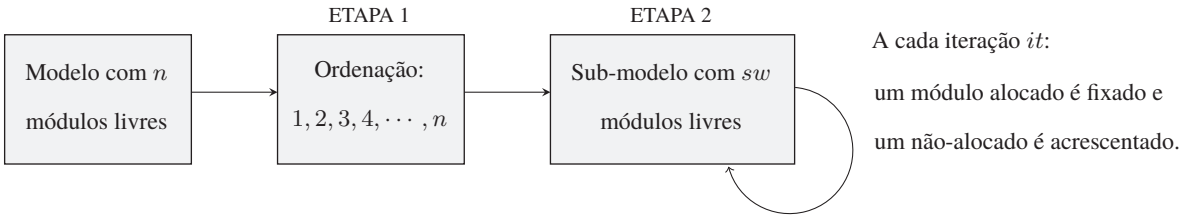


Figura 10 – Etapas da abordagem heurística SW.

Nas Figuras 11 e 12, a abordagem proposta é ilustrada. Na Figura 11, o comportamento do método a cada iteração ($it = 0, \cdots, 4$) é representado visualmente. No exemplo, um conjunto de módulos retangulares, ordenados de 1 a 8, e tamanho da janela deslizante sw igual a 4 são considerados. A cada iteração, o primeiro módulo livre listado tem suas posições relativas fixadas (itens cinzas) e a janela de módulos livres é deslizada para a direita, incluindo o próximo módulo ao sub-problema. Se $(sw + it) = 8$, todos os itens já foram posicionados e então uma solução final é obtida. O *layout* resultante de cada iteração é apresentado na Figura 12. Vale observar, na representação visual de cada iteração na Figura 12, que o retângulo envolvente é ajustado de acordo com a necessidade da alocação feita, assumindo sempre a menor dimensão necessária para conter todos os itens no chip. Ainda, cada módulo livre pode ser incluído entre dois módulos fixos, como visto na Iteração 4, em que o item 8 é posicionado entre os itens 1 e 3.

A Figura 13 apresenta a estrutura geral da abordagem iterativa desenvolvida. As iterações acontecem enquanto existem módulos (itens) não alocados. Caso $(sw + it) = n$, todos os n itens foram considerados e então uma solução final para o modelo original (4.5)-(4.14) é encontrada.

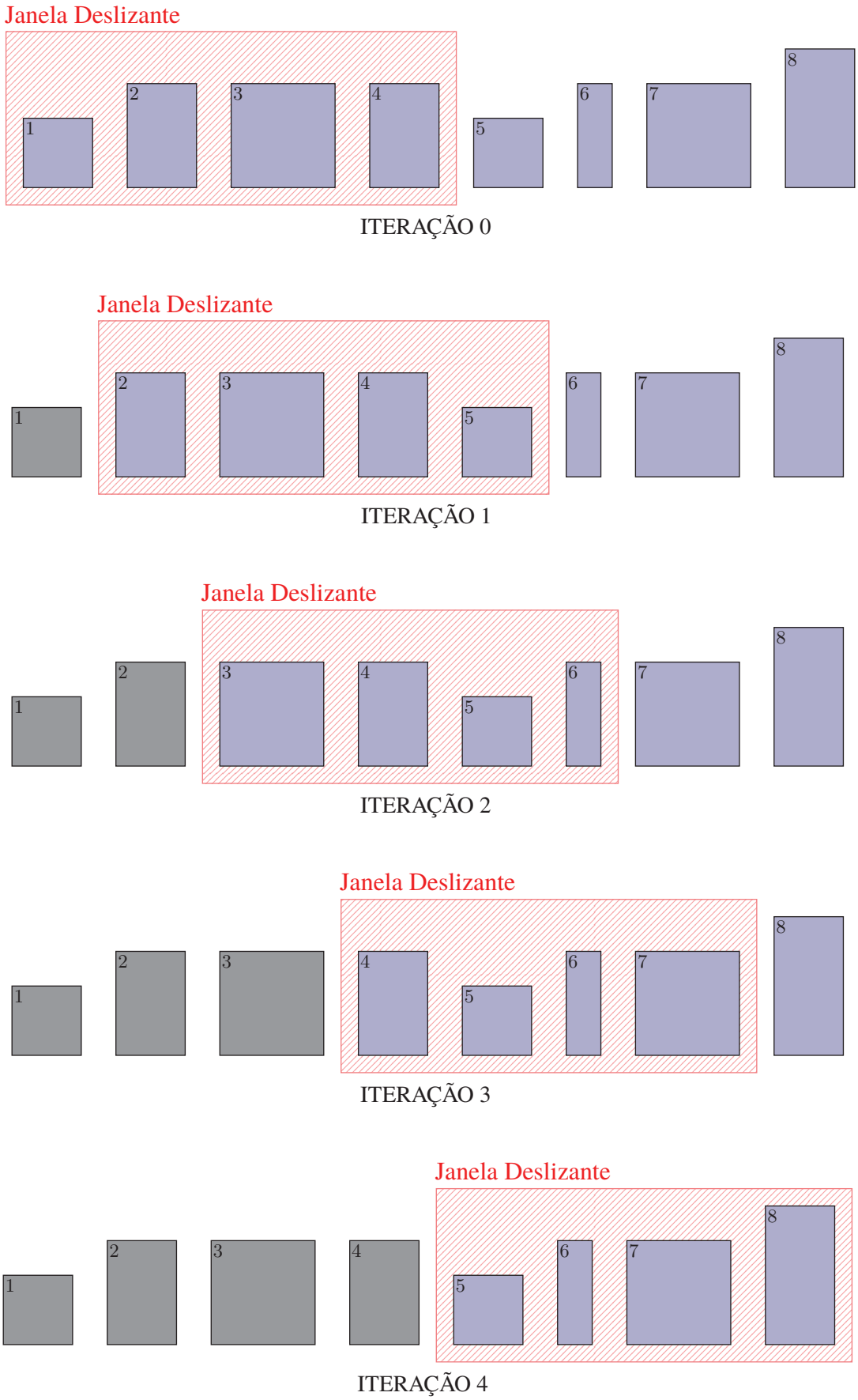
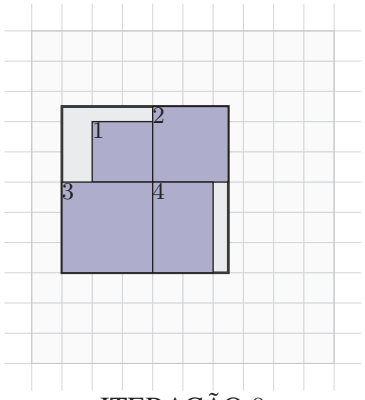


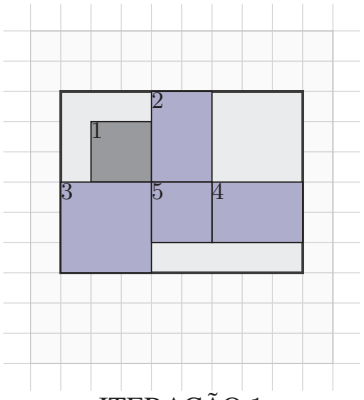
Figura 11 – Exemplo de janela deslizante a cada iteração ($sw = 4$).



ITERAÇÃO 0

Itens livres: $\{1, 2, 3, 4\}$

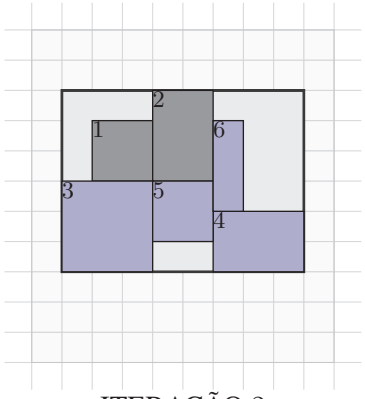
Itens fixados: $\{\}$



ITERAÇÃO 1

Itens livres: $\{2, 3, 4, 5\}$

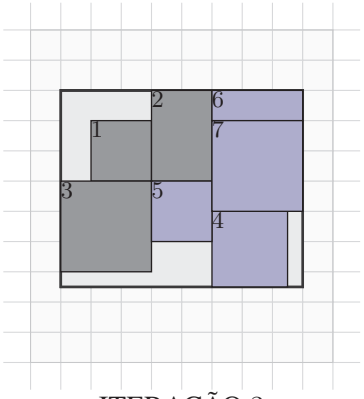
Itens fixados: $\{1\}$



ITERAÇÃO 2

Itens livres: $\{3, 4, 5, 6\}$

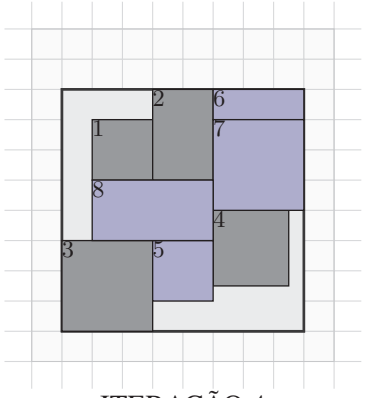
Itens fixados: $\{1, 2\}$



ITERAÇÃO 3

Itens livres: $\{4, 5, 6, 7\}$

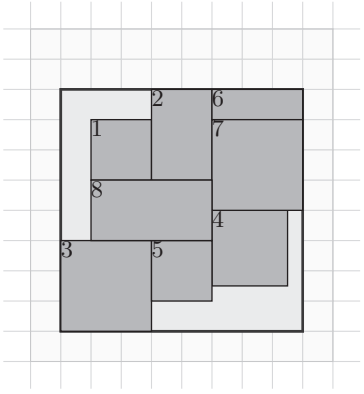
Itens fixados: $\{1, 2, 3\}$



ITERAÇÃO 4

Itens livres: $\{5, 6, 7, 8\}$

Itens fixados: $\{1, 2, 3, 4\}$



LAYOUT FINAL

Todos os itens alocados

$J = \{1, 2, 3, 4\} \cup \{5, 6, 7, 8\}$

Figura 12 – Exemplo de *layout* a cada iteração ($sw = 4$).

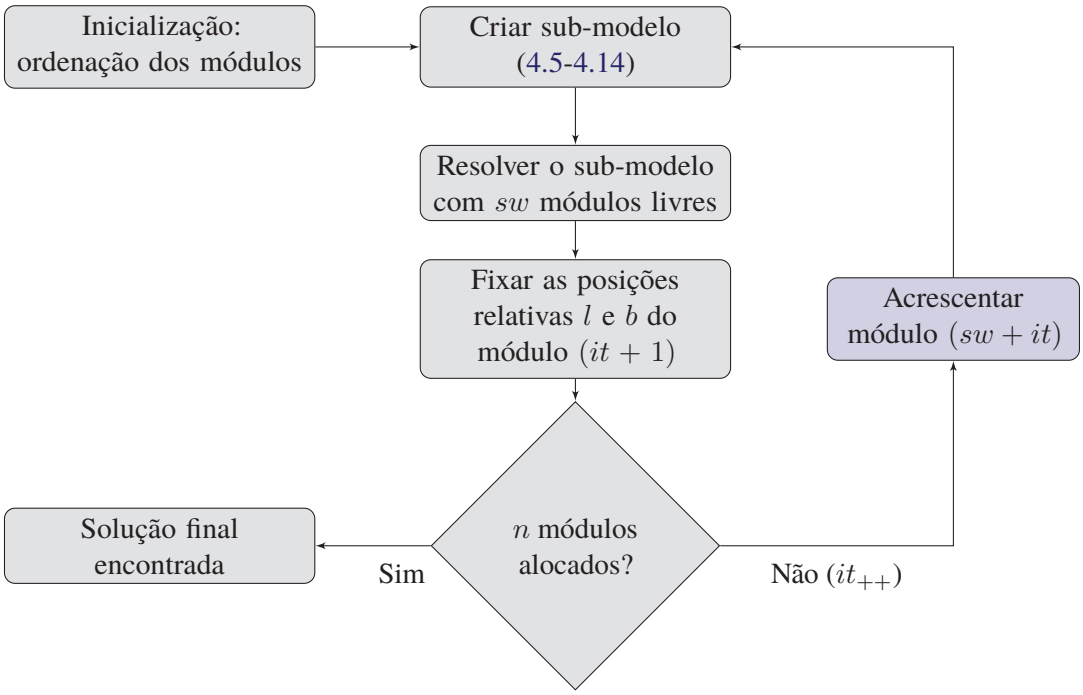


Figura 13 – Estrutura da abordagem heurística SW.

5.2 Mateurística BRKGA-SW

Nesta seção é descrito o algoritmo genético de chave-aleatória tendenciosa (BRKGA) adaptado para o problema de empacotamento de módulos flexíveis em circuitos VLSI. Uma mateurística é um algoritmo de otimização que surge pela integração de meta-heurísticas e programação matemática, assim uma mateurística BRKGA-SW é apresentada nesta Seção. A estrutura geral da mateurística foi revisada no Capítulo 3. Os valores adotados para cada parâmetro são especificados no Capítulo 6.

Em um BRKGA, os indivíduos consistem em um vetor de números reais aleatórios no intervalo de $[0, 1)$, sendo estes números também chamados de chaves. Um decodificador tem como entrada uma população de p indivíduos e retorna uma solução viável para o problema, de modo que seja possível avaliar a função objetivo desta solução. Para descrever um BRKGA para um problema de otimização combinatória específico, basta mostrar como as soluções são codificadas como vetores de chaves aleatórias e como esses vetores são decodificados para soluções viáveis do problema de otimização.

São parâmetros essenciais para a compressão do algoritmo neste trabalho:

- p: tamanho da população;
- p_e: tamanho da população considerada elite;

- $p - p_e$: tamanho da população não-elite;
- p_m: tamanho da população de indivíduos mutantes;
- p_b: probabilidade de herança de um pai elite no cruzamento;
- Critério de parada: tempo computacional ou taxa mínima de melhoria de uma população para outra.

No BRKGA adaptado para o problema de empacotamento de módulos flexíveis em circuitos VLSI, um indivíduo é uma solução do *floorplanning* representada por um vetor de chaves aleatórias que possuem uma relação biunívoca com os módulos a serem posicionados e suas variáveis de posicionamento. Uma população é um conjunto de indivíduos e cada indivíduo tem sua aptidão determinada pela função objetivo (4.5) do modelo proposto na Seção 4. Os indivíduos com valores de função objetivo menores são mais aptos do que aqueles com valores maiores, uma vez que o problema abordado é de minimização. Os indivíduos de maior aptidão, isto é, as soluções com os menores valores de comprimento de fio utilizado nas interconexões dos módulos, compõem o grupo de elite da população.

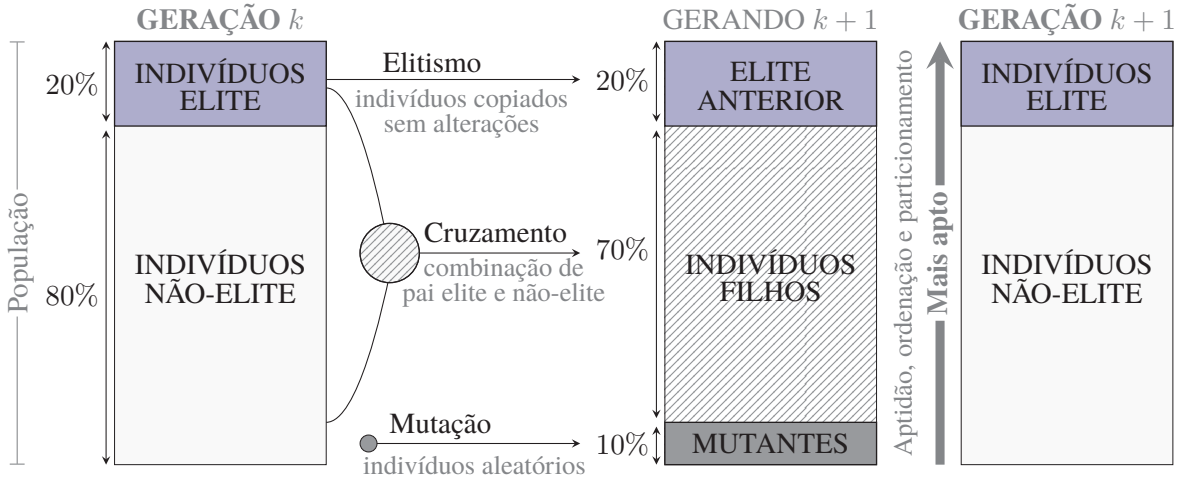


Figura 14 – Esquema geral de processo evolutivo do BRKGA.

O esquema na Figura 14 apresenta parte do processo do BRKGA. O algoritmo começa com a geração inicial k, composta por uma população de p vetores aleatórios de tamanho n ou $2n$, sendo n o número de módulos flexíveis considerados no *floorplanning*. A população é particionada em elite e não-elite de acordo com a aptidão dos indivíduos. A cada iteração uma nova geração é formada com uma nova população, composta pelos p_e indivíduos elite da geração anterior, p_m indivíduos mutantes gerados através da operação de mutação e $p - p_e - p_m$ indivíduos descendentes gerados através da combinação de duas soluções, um pai elite e um pai não-elite. Na iteração seguinte, a população é particionada novamente e todo o processo se repete até que o critério de parada seja atingindo. O algoritmo retorna o indivíduo mais apto

encontrado durante toda a execução, isto é, a solução que assume o menor comprimento de fio na busca realizada.

Além da decodificação feita para adaptar o BRKGA ao problema abordado neste trabalho, uma combinação entre as formulações foi proposta. O BRKGA-SW agrega a janela deslizante ao algoritmo genético na etapa evolutiva de uma geração para a geração seguinte. Ao calcular a aptidão de cada indivíduo da população, o modelo (4.5)-(4.14) é executado através da heurística SW (Seção 5.1) que estrutura a resolução do *floorplanning* utilizando janela deslizante para reduzir a dificuldade e tempo do algoritmo a cada iteração inicialização. Como inicialização, adota-se o padrão do BRKGA de ordenação não-decrescente do vetor real gerado aleatoriamente.

Os resultados obtidos pela heurística SW e pela mateurística BRKGA-SW, para diferentes instâncias testadas, são apresentados no Capítulo 6.

6 Experimentos computacionais

Para avaliar o desempenho dos procedimentos propostos, testes computacionais foram realizados na linguagem de programação C++ com o solver IBM CPLEX Optimization Studio versão 22.1 e executados em um computador com processador Intel Core i7-2600 com 16GB de memória RAM. Foram utilizadas cinco instâncias do conjunto de referência arquivado no *Microelectronics Center of North Carolina (MCNC)*[1] com origem no projeto de circuitos VLSI, no qual todos os itens têm forma retangular flexível e o número de itens a serem posicionados não ultrapassa os 50. Cada instância contém um número de redes, sendo cada rede um conjunto de módulos (itens) interconectados com composição pré-definida. São elas:

- APTE: 9 itens flexíveis e 97 redes;
- XEROX: 10 itens flexíveis e 203 redes;
- HP: 11 itens flexíveis e 83 redes;
- AMI33: 33 itens flexíveis e 123 redes;
- AMI49: 49 itens flexíveis e 408 redes.

Neste capítulo, são apresentadas as configurações específicas adotadas em cada etapa dos testes e os resultados obtidos na execução do modelo matemático proposto, da abordagem heurística SW e da mateurística BRKGA-SW. Por fim, a solução e tempo computacional de cada um dos métodos é resumida e comparadas entre eles.

6.1 Modelo Matemático

Com o objetivo de minimizar as distâncias do comprimento de fio utilizado para interconectar os módulos no chip, o modelo matemático (4.5)-(4.14) foi implementado com tempo de execução limitado à 15 horas, devido à sua alta complexidade. Sem esta limitação, seria necessário um período de tempo impraticável para atingir uma solução exata, inviabilizando o uso do modelo de forma independente.

[1] http://vlsicad.eecs.umich.edu/BK/MCNCbench/SOFT/

Os resultados obtidos são apresentados na Tabela 3, juntamente com a área total do *layout* resultante da alocação, limitante inferior e GAP para cada instância considerada.

Instância	**Valor da solução**	**Área**	**LB**	**GAP**
AMI33	27478	842625	11124,7	59,52%
AMI49	442221	33406576	202890,0	54,12%
APTE	157715	28901632	156138,0	1,00%
HP	36407	2159136	30888,5	15,16%
XEROX	357932	14081873	345928,0	3,35%

Tabela 3 – Soluções obtidas pelo modelo matemático com tempo limite de 15 horas.

O modelo matemático, proposto no Capítulo 4, tem como resultado as posições possivelmente ideais para o conjunto de módulos alocados, suas dimensões, o comprimento de fio HPWL utilizado para interconectá-los de acordo com as redes e o tempo computacional do procedimento. Através dos resultados obtidos, uma solução geométrica pode ser ilustrada para cada alocação obtida. Na Figura 15, por exemplo, é possível visualizar as alocações geradas para as instâncias APTE, AMI49, AMI33 e HP, correspondentes aos dados supracitados na Tabela 3.

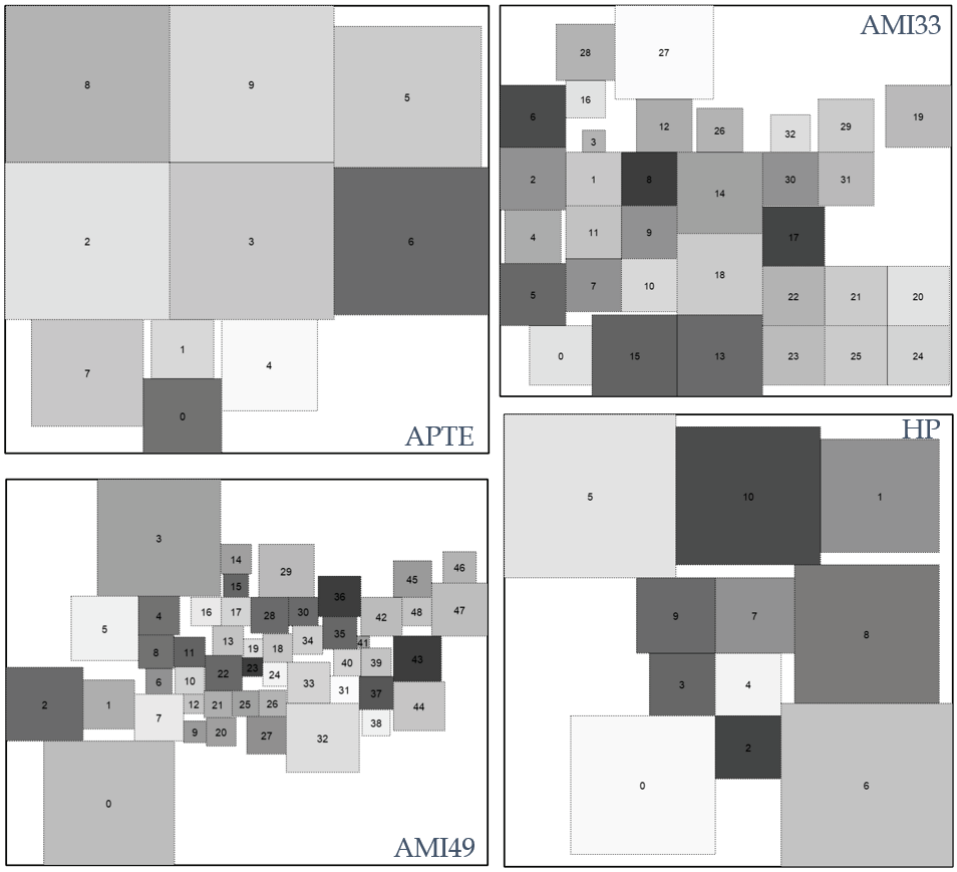


Figura 15 – Alocações (*layouts*) obtidas para as instâncias APTE, AMI49, AMI33 e HP.

6.2 Heurística SW

Nesta seção, os resultados obtidos pela heurística SW são apresentados. As instâncias foram testadas para diferentes valores do parâmetro sw, que corresponde ao tamanho da janela deslizante no modelo e indica a quantidade de variáveis livres a serem consideradas por iteração, variando nos valores $sw = [1, 2, 3, 4, 5]$, sendo 1 o menor valor possível para sw, e os demais valores escolhidos arbitrariamente. Ainda, quatro critérios de ordenação foram considerados na inicialização dos módulos, também chamados de itens: (a) área de modo não-decrescente, (b) área de modo não-crescente, (c) conectividade não-decrescente e (d) conectividade não-crescente. Foi estipulado o limite de tempo de 15 minutos para a obtenção de cada solução no sub-modelo, isto é, cada iteração.

Nas Tabelas 4 a 13 são apresentados os valores de comprimento de fio e tempo computacional, em segundos, obtidos pela heurística para cada instância, combinando diferentes critérios de ordenação inicial e tamanhos de janela. Destaca-se o melhor resultado obtido para a instância e o tempo computacional correspondente.

AMI33	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	30184	27696	27937	25865	25445
Maior área (b)	29694	26334	27346	24973	25347
Menor conectividade (c)	30468	28007	29488	25326	26404
Maior conectividade (d)	27048	25319	26354	25354	**24658**

Tabela 4 – Comprimento de fio obtido por ordenação para a instância AMI33.

AMI33	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	7682	2524	7218	1652	19742
Maior área (b)	2739	1079	2421	6330	15733
Menor conectividade (c)	2834	1968	1419	8537	15629
Maior conectividade (d)	2297	1898	5421	7741	**10532**

Tabela 5 – Tempo computacional obtido por ordenação para a instância AMI33.

Os valores obtidos pelo modelo matemático (4.5)-(4.14), com tempo de execução de 15 horas, foram rapidamente superados pelos resultados encontrados pela heurística SW. Para a instância AMI33, a função objetivo do modelo exato obteve como resultado 27478 em 15 horas, enquanto a heurística SW obteve o resultado 24658, em um tempo computacional de aproximadamente 3 horas. Esses valores indicam que através da abordagem herística proposta é possível atingir boas soluções em um tempo muito menor.

Para a instância AMI49 (Tabela 6), o melhor resultado obtido considerou como o critério de inicialização a conectividade de itens de modo não-crescente, característica que se manteve para as demais instâncias: AMI33 e APTE. Nas Tabelas 10 e 13, observa-se que a melhor solução encontrada para a instância HP foi obtida ao adotar o critério de ordenação por área de modo não-crescente, considerando uma janela deslizante de tamanho 5.

AMI49	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	472657	462784	444773	442852	436985
Maior área (b)	626015	620124	645093	521928	498833
Menor conectividade (c)	550140	597468	528251	518566	510881
Maior conectividade (d)	469879	441883	440818	**413738**	423920

Tabela 6 – Comprimento de fio por ordenação para a instância AMI49.

AMI49	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	8638	13475	8282	20022	25196
Maior área (b)	17929	4957	7528	19489	48636
Menor conectividade (c)	42394	11350	2402	4931	10717
Maior conectividade (d)	32351	8307	3458	**7541**	20483

Tabela 7 – Tempo computacional obtido por ordenação para a instância AMI49.

APTE	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	159344	173529	172347	165792	158001
Maior área (b)	162859	163930	159134	159134	157718
Menor conectividade (c)	183876	169328	170968	159141	159252
Maior conectividade (d)	162860	164610	157739	159245	**157718**

Tabela 8 – Comprimento de fio por ordenação para a instância APTE.

APTE	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	819	319	221	127	156
Maior área (b)	110	246	713	315	115
Menor conectividade (c)	600	261	638	211	100
Maior conectividade (d)	598	262	399	277	**923**

Tabela 9 – Tempo computacional por ordenação para a instância APTE.

HP	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	41010	41259	38553	37588	37083
Maior área (b)	42034	41194	36681	36681	**36659**
Menor conectividade (c)	43383	40018	39228	39017	37692
Maior conectividade (d)	38847	37252	36832	36832	36832

Tabela 10 – Comprimento de fio por ordenação para a instância HP.

HP	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	1303	3595	1376	392	767
Maior área (b)	1317	2518	631	2190	**1904**
Menor conectividade (c)	1414	4083	1205	1491	3235
Maior conectividade (d)	1397	3544	1713	795	986

Tabela 11 – Tempo computacional por ordenação para a instância HP.

XEROX	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	374661	386350	377546	375599	360550
Maior área (b)	367020	359104	357935	357932	**357924**
Menor conectividade (c)	385671	386482	380994	367024	372432
Maior conectividade (d)	412978	373382	375104	371501	362655

Tabela 12 – Comprimento de fio por ordenação para a instância XEROX.

XEROX	$sw = 1$	$sw = 2$	$sw = 3$	$sw = 4$	$sw = 5$
Menor área (a)	130	438	234	182	605
Maior área (b)	140	373	91	329	**224**
Menor conectividade (c)	712	353	127	86	1040
Maior conectividade (d)	706	492	115	106	423

Tabela 13 – Tempo computacional por ordenação para a instância XEROX.

Ao definir um critério de ordenação inicial é possível estabelecer quais itens serão priorizados na alocação feita e, com isso, melhorar o processo iterativo realizado. De acordo com os dados supracitados nas Tabelas 4 a 13, para todas as instâncias testadas, os melhores resultados foram obtidos ao utilizar ordenações não-crescentes como estratégia inicial e, na maioria dos casos, ordenações baseadas nas características da descrição detalhada da conectividade do VLSI.

As soluções obtidas indicam grande influência do critério de ordenação no resultado final do problema. Com tendência a obter o melhor resultado, ordenar os itens pela conectividade de modo não-crescente garante que o processo de alocação seja iniciado fixando primeiramente os itens que possuem maior conectividade e flexibilizando o posicionamento dos itens que aparecem em um número menor de conexões. Nas Figuras 16 e 17 é possível visualizar a solução geométrica (*layouts*) de algumas alocações obtidas para a instância AMI33 com diferentes tamanhos de janela e ordenações por conectividade.

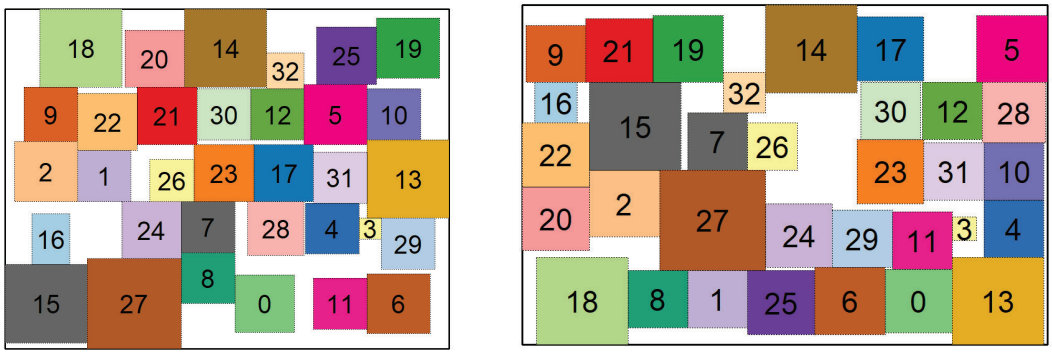


Figura 16 – Alocações obtidas para AMI33: itens ordenados por menor conectividade ($sw = 3$ e $sw = 5$, respectivamente).

Com as soluções geométricas dadas, nota-se que os *layouts* possuem espaço não utilizado (espaço morto) no interior do retângulo envolvente e os módulos flexíveis foram ajustados

predominantemente em formato quadrangular, em que $w_i = h_i$ e $s_i = 1$. Em nenhum dos testes considerados para as cinco instâncias, o número de itens flexionados com $s_i \neq 1$ atinge valor maior que 30%. Este resultado indica que a característica de flexibilidade dos itens foi pouco explorada no espaço de soluções. A tendência é que alocações com itens flexíveis sejam mais compactas, isto é, tenham espaço morto reduzido, além de menores valores de comprimento de fio utilizado nas interconexões, objetivo principal do problema.

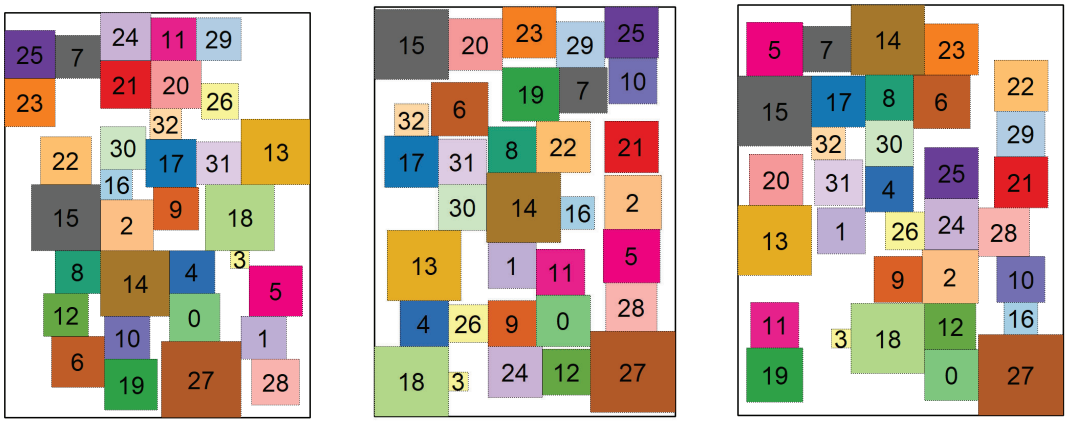


Figura 17 – Alocações obtidas para AMI33: itens ordenados por maior conectividade ($sw = 1$, $sw = 3$ e $sw = 5$, respectivamente).

Na Figura 18 a comparação de soluções entre as instâncias para cada critério de ordenação é ilustrada graficamente. No eixo vertical, é expresso o percentual que indica a distância da solução em relação à melhor solução obtida para aquela instância. No eixo horizontal, cada instância é indicada pelo seu nome seguindo, respectivamente, o tamanho de janela $sw = [1, 2, 3, 4, 5]$. O critério que prioriza itens com maior conectividade fica em destaque, com valores inferiores aos demais critérios nas instâncias AMI33, AMI49, APTE e HP.

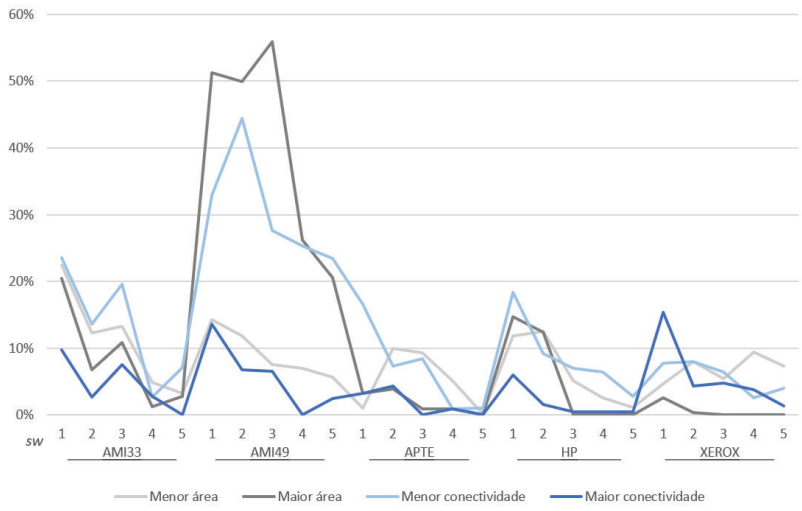


Figura 18 – GAP de soluções entre instâncias por critério de ordenação.

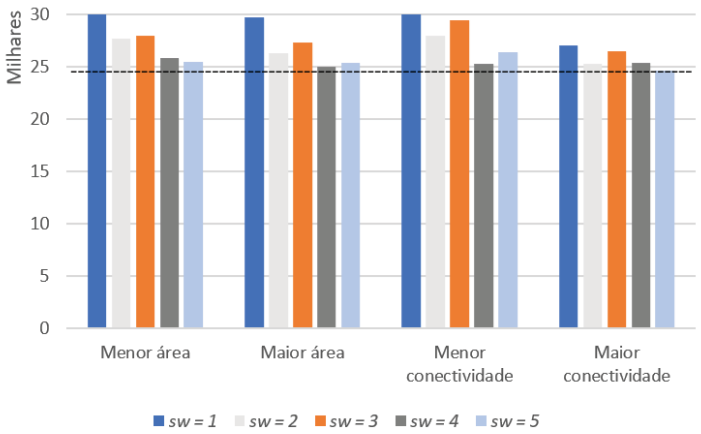


Figura 19 – Comprimento de fio por ordenação para a instância AMI33.

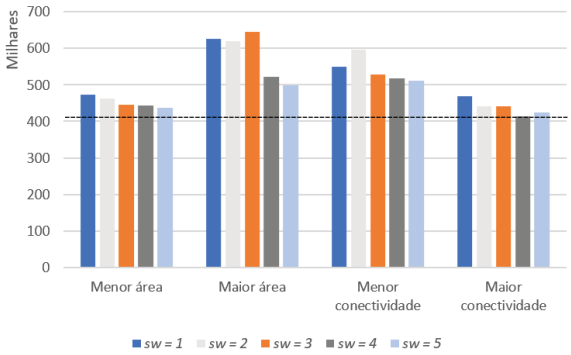


Figura 20 – Solução para AMI49.

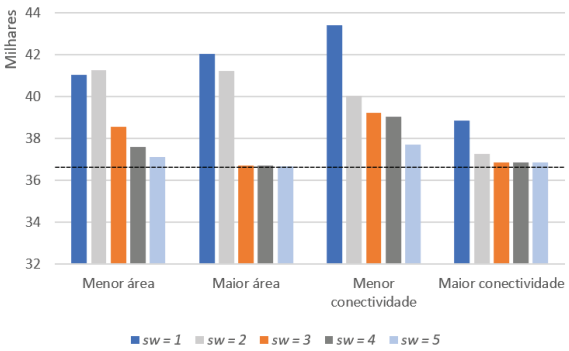


Figura 21 – Solução para HP.

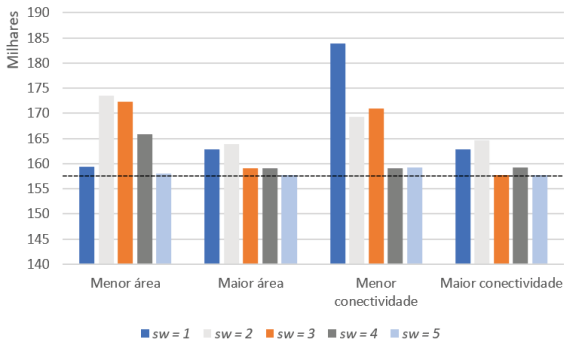


Figura 22 – Solução para APTE.

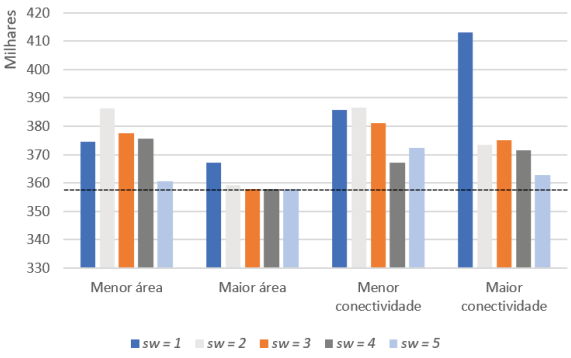


Figura 23 – Solução para XEROX.

Ainda, vale observar a relação entre o tempo computacional e o tamanho de janela adotado. Como proposto na abordagem, ao criar sub-modelos com uma quantidade limitada de variáveis que podem ser posicionadas livremente, a dificuldade de resolução é reduzida e o tempo de execução diminui consideravelmente. Tamanhos de janelas maiores tendem a gerar resultados melhores, porém com tempos de processamento muito mais elevados em alguns casos, nem sempre viáveis. Por exemplo, na Tabela 4, para a instância AMI33 com ordenação por conectividade não-decrescente, a solução para janela de tamanho 5 é cerca de 11% melhor do que a solução para janela 3, porém o tempo computacional para a janela 5 é 10 vezes maior em relação à janela 3.

É possível observar pelo conjunto de gráficos referentes ao comprimento de fio obtido para cada instância (Figuras 19 a 23), que existe um salto significativo entre os resultados de uma estratégia de inicialização para outra. Tal característica pode sugerir que o critério de ordenação adotado tem maior influência na qualidade da solução do que o tamanho da janela escolhida, nos casos em que sw é maior que 3. Ao calcular a razão entre a função objetivo dos testes de janela deslizante com tamanho 3 e 5, pode-se notar que em apenas 10% dos testes foi encontrada uma melhora superior a 10% ao aumentar o tamanho da janela, isto é, em 90% das execuções a diferença entre os resultados obtidos é pequena.

Na Figura 24 é ilustrada a comparação de soluções entre as instâncias para cada tamanho de janela considerado. No eixo vertical, é expresso o percentual que indica a distância da solução atual em relação à melhor solução obtida para aquela instância. No eixo horizontal, cada instância é indicada pelo seu nome seguindo, respectivamente, o critério de ordenação adotado: área não-decrescente, área não-crescente, conectividade não-decrescente e conectividade não-crescente, respectivamente.

Na Figura 25 é ilustrada a comparação de valores do tempo computacional entre as instâncias para cada tamanho sw de janela deslizante. No eixo vertical, é expresso o percentual que indica a distância do tempo de processamento atual em relação ao melhor tempo obtido para aquela instância. No eixo horizontal, cada instância é indicada de forma análoga à Figura 24. Os resultados para as janelas de tamanho 3 e 5 são discrepantes na maioria dos casos considerados.

O alto índice de elevação do tempo computacional ao alterar o tamanho da janela deslizante, sem que haja aumento proporcional na solução obtida, pode ser resultado das definições prévias de execução dos testes. Com o limite de tempo de 900 segundos, estipulado inicialmente para a obtenção de cada solução no sub-modelo, não foi possível encontrar boas soluções nas iterações. Devido à complexidade do problema ao adotar mais itens com variáveis de posicionamento livres, estima-se que seja necessário uma quantidade maior de tempo por iteração para que melhores alocações sejam obtidas. A descrição das instâncias está apresentada no Apêndice.

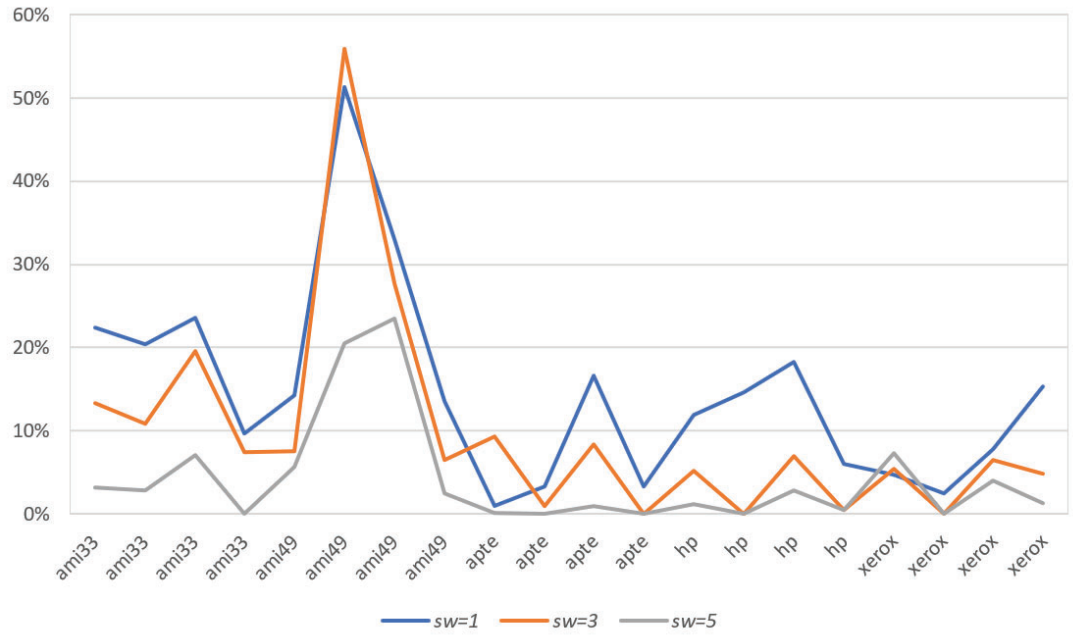


Figura 24 – GAP de soluções obtidas entre instâncias por tamanho de janela deslizante sw.

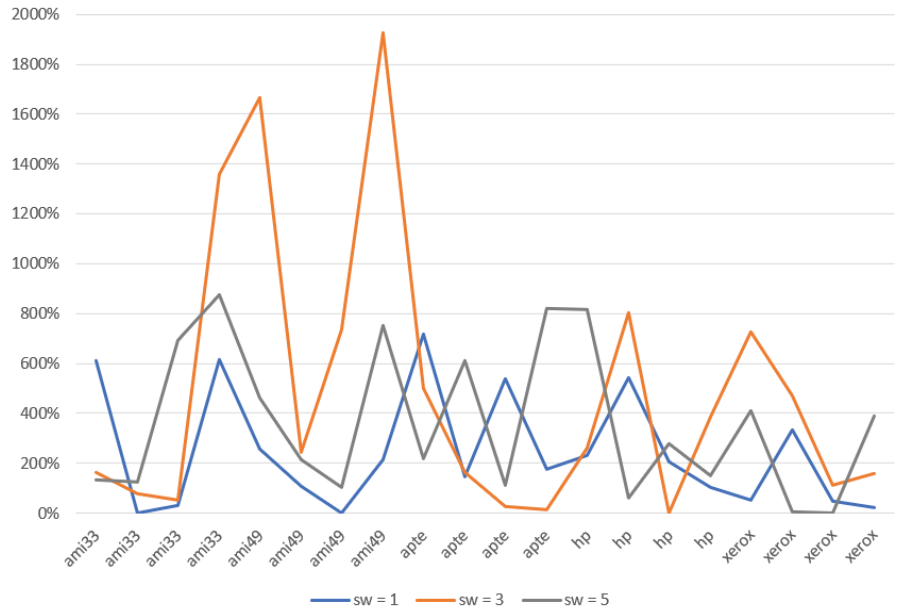


Figura 25 – GAP de valores de tempo computacional registrado entre instâncias por tamanho de janela deslizante sw.

6.3 BRKGA-SW

Nesta seção, os resultados obtidos BRKGA-SW são apresentados. Para facilidade de notação, denotamos BRKGA a formulação com $sw = n$, isto é, sem o uso efetivo da janela deslizante. Para implementar o algoritmo, os valores dos parâmetros utilizados nos experimentos foram baseados nas recomendações de Gonçalves e Resende (2011). A população p assume o valor $2n$ para o BRKGA e n para o BRKGA-SW, sendo n o número de módulos flexíveis. Para a

população elite e mutante, é dado $p_e = (0{,}2 \cdot p)$ e $p_m = (0{,}1 \cdot p)$, respectivamente, e probabilidade de herança de um pai elite no cruzamento $p_b = [0{,}5; 0{,}7]$. Os valores para p_b foram definidos de modo a avaliar os efeitos gerados na solução ao tendenciar a herança no cruzamento. Quando $p_b = 0{,}5$ o algoritmo genético assume que ambos os pais (elite e não-elite) têm a mesma chance no processo de cruzamento, enquanto $p_b = 0{,}7$ garante a tendência de escolha das características do pai elite, conforme previsto na formulação clássica do BRKGA. Os resultados obtidos para o BRKGA e BRKGA-SW são apresentados nas Tabelas 14 a 15. Vale observar que cada iteração é iniciada antes de atingir o tempo limite, mas sua execução é feita até o fim, assim podem atingir tempos computacionais superiores à 1 hora.

	$p_b = 0{,}5$			$p_b = 0{,}7$		
INSTÂNCIA	**SOLUÇÃO**	**ITERAÇÕES**	**TEMPO**	**SOLUÇÃO**	**ITERAÇÕES**	**TEMPO**
AMI33	46963	203	3616	48003	214	3607
AMI49	724772	164	3613	779779	168	3600
APTE	177216	105	3601	177216	1096	3601
HP	50481	1089	3600	43113	1126	3602
XEROX	421574	394	1783	435516	10	52

Tabela 14 – Soluções BRKGA com tempo limite de execução (critério de parada) de 1 hora.

	$sw = 1$			$sw = 2$		
INSTÂNCIA	**SOLUÇÃO**	**ITERAÇÕES**	**TEMPO**	**SOLUÇÃO**	**ITERAÇÕES**	**TEMPO**
AMI33	26288	1	11312	25452	1	107990
AMI49	438688	1	11607	416377	1	141373
APTE	157715	71	3605	157715	17	3651
HP	36659	40	3601	36407	12	3602
XEROX	363636	43	3649	357930	12	3724

Tabela 15 – Soluções BRKGA-SW e tempo limite de execução (critério de parada) de 1 hora.

Os experimentos feitos com o BRKGA (Tabela 14) atingem uma quantidade muito maior de iterações quando comparadas ao BRKGA-SW, isto é, no BRKGA a população evolui mais rapidamente. Apesar disso, mesmo com a tendência de herança do pai elite na operação de cruzamento, as soluções convergem de forma lenta. Quando comparados os casos com alteração no valor de p_b, não há garantia de que a qualidade da solução seja maior ao aumentar p_b. Este é um resultado esperado, dado que o parâmetro p_b indica uma probabilidade.

Ao inserir a estratégia de janela deslizante no BRKGA, uma melhoria significativa foi obtida nas soluções. O BRKGA-SW obtém resultados que indicam maior aprofundamento na evolução de uma geração para outra e, desta forma, o método atinge soluções melhores com uma quantidade menor de iterações necessárias em um período de tempo similar. Para a instância HP, por exemplo, usando o método BRKGA-SW foi encontrada a solução 36407 em apenas 12 iterações com $p_b = 0{,}7$ e tempo limitado à 1 hora. Nestas mesmas condições, no BRKGA foi obtida a solução 43113 em 1126 iterações.

	SW		BRKGA		BRKGA-SW		MODELO	
INSTÂNCIA	**SOLUÇÃO**	**TEMPO**	**SOLUÇÃO**	**TEMPO**	**SOLUÇÃO**	**TEMPO**	**SOLUÇÃO**	**TEMPO**
AMI33	**24658**	10532	48003	3607	25452	107990	27479	54019
AMI49	413738	7541	779779	3613	**416377**	141373	442221	54000
APTE	157718	923	177216	3601	**157715**	3605	157715	11501
HP	36659	1904	43113	3602	**36407**	3602	36407	54008
XEROX	357924	224	435516	1783	**357930**	3724	357932	54008

Tabela 16 – Melhor solução e tempo computacional obtida por cada método proposto.

Nas Figuras 26 e 27, são apresentados os gráficos das soluções e tempo computacional, respectivamente, para os métodos BRKGA, BRKGA-SW1 com $sw = 1$ e BRKGA-SW2 com $sw = 2$. Os resultados obtidos pelo BRKGA-SW apresentam, em média, soluções 65% melhores e encontradas com 5% do número de iterações em relação ao BRKGA, nas mesmas condições e critério de parada. Para as instâncias AMI33 e AMI49, o tempo de processamento de cada iteração é maior no BRKGA-SW, ainda assim, resolvido em tempo viável.

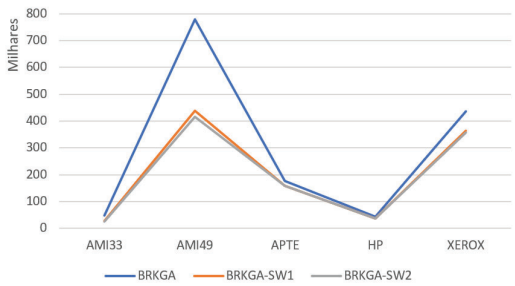


Figura 26 – Soluções dos BRKGAs.

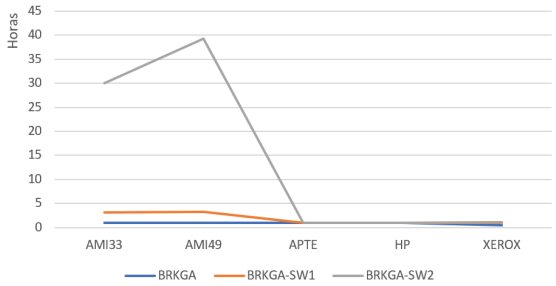


Figura 27 – Tempo dos BRKGAs.

Com os experimentos realizados nas mesmas condições para os três métodos propostos, é possível comparar o desempenho de cada formulação em relação ao comprimento de fio obtido e ao tempo computacional utilizado. Na Tabela 6.3 os melhores resultados obtidos nos testes computacionais são exibidos. Para as cinco instâncias consideradas, é apresentada a melhor solução obtida através da mateurística BRKGA, da adaptação BRKGA-SW (Seção 5.2), da heurística SW (Seção 5.1) e também do modelo matemático (Seção 4) proposto, juntamente com o tempo de execução correspondente à solução. Destaca-se a melhor solução geral obtida.

Na Figura 28 a melhor solução obtida por cada uma das abordagens propostas é apresentada graficamente. Na Figura 29 o tempo computacional correspondente destas soluções é apresentado. Ambos os gráficos foram agrupados por instância.

Como este trabalho aborda um problema de minimização, os menores resultados de comprimento de fio são os mais adequados para representar uma boa solução. Analisando a Figura 28, é possível notar o contraste entre as soluções de cada método. Em cada conjunto de soluções agrupadas por instância existe um padrão nos métodos das melhores e piores soluções obtidas: o método BRKGA atinge a pior solução em todas as instâncias, já as melhores soluções

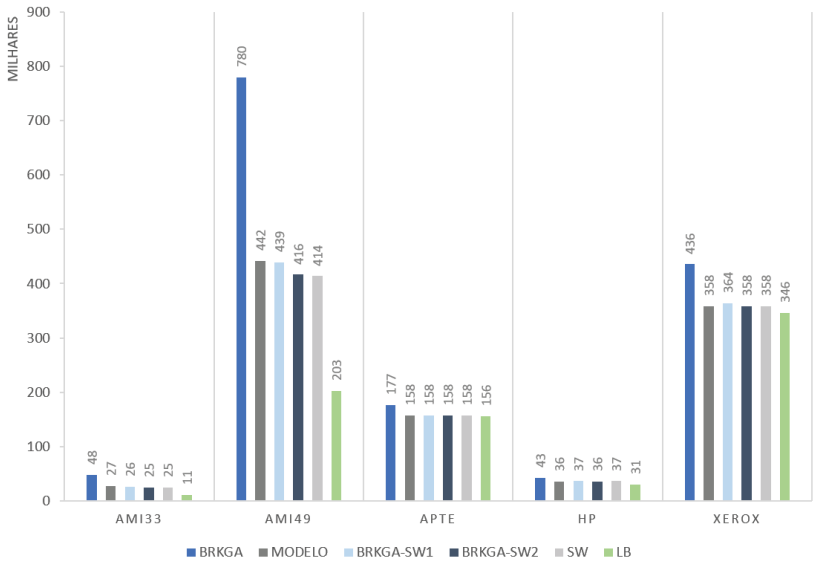


Figura 28 – Melhor solução de cada método nas instância.

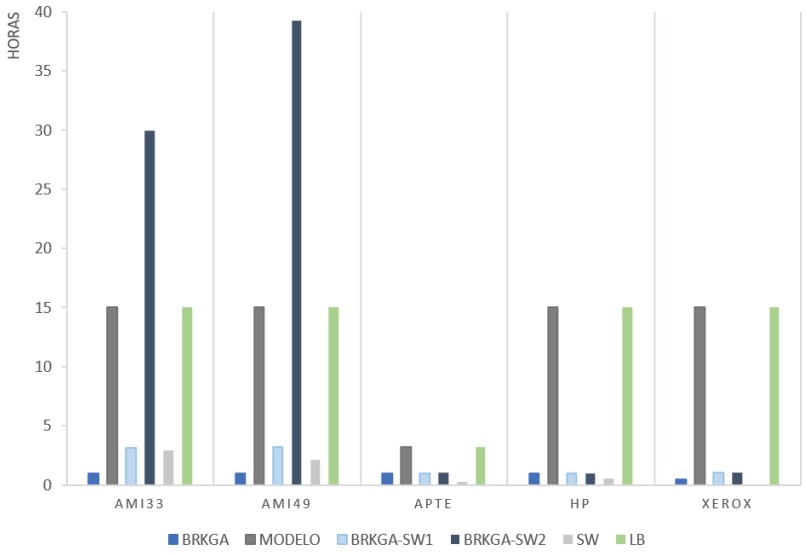


Figura 29 – Tempo computacional da melhor solução de cada método nas instâncias.

foram obtidas pelos métodos que possuem a estratégia de janela deslizante na sua formulação. Assim, a heurística SW e o BRKGA-SW apresentaram melhor desempenho.

A mateurística BRKGA-SW supera o valor obtido pelo modelo matemático em um tempo médio reduzido em 90%. Em relação aos métodos BRKGA independente, o BRKGA-SW utiliza tempo computacional maior em alguns dos cenários. Ainda assim, o tempo pouco mais elevado da abordagem BRKGA-SW é admissível, dado que há uma melhoria significativa nos resultados em um tempo médio reduzido em 85% em relação ao modelo matemático, com $sw = 1$. As soluções obtidas pela heurística SW superam as soluções dos demais métodos com redução de até 48,6% na função objetivo, se aproximando do LB estimado nas instâncias com menor quantidade de itens.

7 Conclusão

Neste trabalho, o problema de planejamento de circuitos VLSI (*floorplanning*) foi abordado como um problema de empacotamento de retângulos flexíveis. Um modelo matemático, uma abordagem heurística e uma mateurística BRKGA foram propostos para a obtenção de soluções para o *floorplanning* quando os módulos a serem alocados possuem dimensões flexíveis e com objetivo de minimizar o comprimento de fio utilizado na interconexão dos módulos.

A heurística SW proposta conta com uma ordenação inicial dos módulos que se deseja alocar e a criação de sub-modelos a cada iteração utilizando uma estratégia de janela deslizante. Um método baseado na meta-heurística BRKGA também foi proposto. O BRKGA foi escolhido como base para a formulação por ser facilmente adaptável, relativamente recente e apresentar bons resultados na literatura em diferentes aplicações. A mateurística BRKGA-SW combina a estrutura genética evolutiva tendenciosa à estratégia de janela deslizante.

A implementação dos procedimentos heurísticos combinados com o modelo matemático gerou bons resultados. As soluções obtidas pelas abordagens propostas, a heurística SW e o BRKGA-SW, superaram rapidamente a qualidade dos valores encontrados pelo modelo matemático isolado. Com a estratégia de inicialização e janela deslizante, utilizada em ambas as abordagens, foi possível atingir um bom conjunto de soluções em um tempo computacional muito menor. Os melhores resultados em todos os testes realizados foram obtidos pelos métodos SW e BRKGA-SW, com soluções reduzidas em até 48% em relação aos demais métodos, incluindo o BRKGA clássico (GONÇALVES; RESENDE, 2011).

Testes computacionais foram realizados com instâncias amplamente utilizadas na literatura, na linguagem de programação C++ com o solver IBM CPLEX Optimization Studio na versão 22.1 e indicam a eficiência do método apresentado. Os resultados demonstram que bons critérios de inicialização têm grande influência sobre a qualidade da solução e estão intimamente ligados às características de conectividade do circuito. Explorar melhores conjuntos de ordenação inicial pode economizar tempo e processamento durante a execução da abordagem, de forma simples e pouco custosa. Além da ordenação pré-estabelecida na heurística SW, o BRKGA-SW gerou boas soluções e mostrou-se robusto.

Apesar do desempenho bastante satisfatório das abordagens, a característica de flexibilidade dos itens foi pouco explorada. Com a possibilidade de ajustar as dimensões dos módulos a serem alocados, melhores configurações podem ser obtidas com *layouts* compactos e compri-

mento de fio reduzido. Tratar a largura e altura dos módulos como variáveis de decisão no modelo matemático influencia na complexidade computacional do problema e tem grande potencial de gerar bons resultados, desta forma esta característica pode certamente ser melhor aproveitada.

Ao estruturar o problema abordado, foi adotada uma simplificação de cálculo que aproxima a medida de comprimento de fio, considerando os pontos de conexão do circuito integrado posicionados no centro de cada um dos módulos. Tal aproximação fornece a distância de forma eficiente, porém é possível calcular o comprimento de fio de forma ainda mais precisa quando as posições exatas dos pontos de conexão são conhecidas. Ao especificar cada ponto de conexão do circuito, a flexibilidade dos itens deve, consequentemente, ser melhor explorada.

Como continuidade deste trabalho, sugere-se utilizar as posições exatas dos pontos de conexão do circuito, incentivar o ajuste de dimensão dos itens flexíveis através de uma modificação na função objetivo ou implementar uma estratégia de finalização na busca de lapidar a solução encontrada para ajustar a área total da região alocada. Ainda, o algoritmo BRKGA-SW pode ser utilizado em diferentes contextos com pequenas adaptações.

Referências

ANAND, S.; SARAVANASANKAR, S.; SUBBARAJ, P. A multiobjective optimization tool for very large scale integrated nonslicing floorplanning. *International Journal of Circuit Theory and Applications*, v. 41, n. 9, p. 904–923, 2013. 2, 15

BAKER, B. S.; COFFMAN JR, E. G.; RIVEST, R. L. Orthogonal packings in two dimensions. *SIAM Journal on computing*, SIAM, v. 9, n. 4, p. 846–855, 1980. 1

BEAN, J. Genetic algorithms and random keysfir sequencing and optimization. *ORSA J. on Computing*, v. 6, 1994. 12

BEZERRA, V. M. R. *Problemas de empacotamento bidimensional em níveis: estratégias baseadas em modelagem matemática*. Tese (Doutorado) — Tese de Doutorado, USP, 2018. 6

BORTFELDT, A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, v. 172, n. 3, p. 814–837, 2006. 11

BORTFELDT, A. A reduction approach for solving the rectangle packing area minimization problem. *European Journal of Operational Research*, Elsevier, v. 224, n. 3, p. 486–496, 2013. 1, 11

BUI, Q. T.; VIDAL, T.; HÀ, M. H. On three soft rectangle packing problems with guillotine constraints. *Journal of global optimization*, Springer, v. 74, n. 1, p. 45–62, 2019. 11

BURIOL, L. et al. A biased random-key genetic algorithm for road congestion minimization. *Optimization Letters*, v. 4, n. 4, p. 619–633, 2010. 13

CHAN, K.; HSU, C.; LIN, J. A flexible fixed-outline floorplanning methodology for mixed-size modules. *Asia and South Pacific Design Automation Conference (ASP-DAC)*, v. 18, 2013. 2, 15, 20

CHEN, D.-S. et al. Fixed-outline floorplanning using robust evolutionary search. *Engineering Applications of Artificial Intelligence*, v. 20, n. 6, p. 821–830, 2007. 15

CHEN, J.; ZHU, W.; ALI, M. A hybrid simulated annealing algorithm for nonslicing vlsi floorplanning. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, v. 41, n. 4, p. 544–553, 2011. 2, 15

CHEN, T.-C.; CHANG, Y.-W. Floorplanning. In: *Electronic Design Automation*. [S.l.]: Elsevier, 2009. p. 575–634. 1, 5, 15, 17, 18, 19, 20, 21, 25

CHERRI, A. C. *Algumas extensões do problema de corte de estoque com sobras de material aproveitáveis*. Tese (Doutorado) — Tese de Doutorado, USP, 2009. 11

CHI, J.; CHI, M. An effective soft module floorplanning algorithm based on sequence pair. *IEEE International ASIC/SOC Conference*, v. 15, 2002. 15

ERICSSON, M.; RESENDE, M.; PARDALOS, P. A genetic algorithm for the weight setting problem in ospf routing. *Journal of Combinatorial Optimization*, v. 6, n. 3, p. 299–333, 2002. 2, 13

FEKETE, S. P.; SCHEPERS, J. On more-dimensional packing iii: Exact algorithms. Citeseer, 2000. 1

GONÇALVES, J. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European Journal of Operational Research*, v. 183, p. 1212–1229, 02 2007. 13

GONÇALVES, J.; MENDES, J. D. M.; RESENDE, M. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, v. 167, n. 1, p. 77–95, 2005. 2, 13

GONÇALVES, J.; RESENDE, M. An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering*, v. 47, n. 2-3, p. 247–273, 2004. 13

GONÇALVES, J. F.; RESENDE, M. G. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, Springer, v. 17, n. 5, p. 487–525, 2011. 13, 14, 42, 46

GRACIA, D.; RAJARAM, S. A novel differential evolution based optimization algorithm for non-sliceable vlsi floorplanning. *Iranian Journal of Science and Technology, Transaction A: Science*, v. 39, p. 375–382, 2015. 15

GUIMARãES, M. et al. A biased random-key genetic algorithm for the 2-dimensional guillotine cutting stock problem with stack constraints. *Communications in Computer and Information Science*, v. 1541 CCIS, p. 155–169, 2022. 14

HE, K.; JI, P.; LI, C. Dynamic reduction heuristics for the rectangle packing area minimization problem. *European Journal of Operational Research*, v. 241, n. 3, p. 674–685, 2015. 1, 2, 15

HOLLAND, J. H. *Adaptation in natural and artificial systems*. [S.l.]: University of Michigan Press, 1975. 1ª edição. 2ª edição em 1992. 12

HUANG, W.; CHEN, D.; XU, R. A new heuristic algorithm for rectangle packing. *Computers & Operations Research*, Elsevier, v. 34, n. 11, p. 3270–3280, 2007. 1

IBARAKI, T.; NAKAMURA, K. Packing problems with soft rectangles. In: SPRINGER. *International Workshop on Hybrid Metaheuristics*. [S.l.], 2006. p. 13–27. 10

IMAHORI, S.; YAGIURA, M.; IBARAKI, T. Local search algorithms for the rectangle packing problem with general spatial costs. *Mathematical Programming*, v. 3, n. 97, p. 542–569, 2003. 11

IMAHORI, S.; YAGIURA, M.; IBARAKI, T. Improved local search algorithms for the rectangle packing problem with general spatial costs. *European Journal of Operational Research*, v. 3, n. 167, p. 48–67, 2005. 11

IORI, M. et al. Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, v. 289, n. 2, p. 399 – 415, 2021. 10, 11

JENIFER, J.; ANAND, S.; LEVINGSTAN, Y. Simulated annealing algorithm for modern vlsi floorplanning problem. *ICTACT Journal on microelectronics*, v. 2, 2016. 16, 21

JI, P. et al. An iterative merging algorithm for soft rectangle packing and its extension for application of fixed-outline floorplanning of soft modules. *Computers & Operations Research*, Elsevier, v. 86, p. 110–123, 2017. 16

KAHNG, A. B. et al. Global and detailed placement. In: *VLSI Physical Design: From Graph Partitioning to Timing Closure*. [S.l.]: Springer, 2011. p. 95–126. 1, 5, 20, 21

LASKAR, N. et al. A survey on vlsi floorplanning: Its representation and modern approaches of optimization. *IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems*, 2015. 15

LAUDIS, L.; RAMADASS, N. A lion's pride inspired algorithm for vlsi floorplanning. *Journal of Circuits, Systems and Computers*, v. 29, n. 1, 2020. 16

LIMA, A. et al. A multi-population brkga for the automatic clustering problem. In: . [S.l.: s.n.], 2021. p. 368–373. 14

LODI, A.; MONACI, M. Integer linear programming models for 2-staged two-dimensional knapsack problems. *Mathematical Programming, Series B*, v. 94, n. 2-3, p. 257 – 278, 2003. 10

MAINIERI, G. B. *Meta-heurística BRKGA aplicada a um problema de programação de tarefas no ambiente flowshop híbrido*. Tese (Doutorado), 2014. USP. Orientação: Ronconi, D. P. 2, 13

MARTELLO, S.; MONACI, M. Models and algorithms for packing rectangles into the smallest square. *Computers & Operations Research*, Elsevier, v. 63, p. 161–171, 2015. 1

MAURI, G. et al. Hybrid metaheuristics to solve a multiproduct two-stage capacitated facility location problem. *International Transactions in Operational Research*, v. 28, n. 6, p. 3069–3093, 2021. 2, 14

MURATA, H. et al. Vlsi module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on Computer*, Aided Design, p. 1518–1524, 1996. 20

MURATA, H.; KUH, E. S. Sequence-pair based placement method for hard/soft/pre-placed modules. In: ACM. *Proceedings of the 1998 international symposium on Physical design*. [S.l.], 1998. p. 167–172. 10

MöNCH, L.; ROOB, S. A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Applied Soft Computing Journal*, v. 68, p. 835–846, 2018. 2, 14

NORONHA, T. F.; RESENDE, M. G. C.; RIBEIRO, C. C. A biased random-key genetic algorithm for routing and wavelength assignment. *Journal of Global Optimization*, v. 50, n. 3, p. 503 – 518, 2011. 2, 13

ORTMANN, F. G.; VUUREN, J. H. van. Modified strip packing heuristics for the rectangular variable-sized bin packing problem. *ORiON*, v. 1, n. 26, p. 21 – 44, 2010. 11

PRASETYO, H. et al. Survey on applications of biased-random key genetic algorithms for solving optimization problems. In: . [S.l.: s.n.], 2016. v. 2016-January, p. 863–870. 14

RESENDE, M. et al. A biased random-key genetic algorithm for the steiner triple covering problem. *Optimization Letters*, v. 6, n. 4, p. 605–619, 2012. Cited By 16. 13

SHUNMUGATHAMMAL, M.; COLUMBUS, C.; ANAND, S. A nature inspired optimization algorithm for vlsi fixed-outline floorplanning. *Analog Integrated Circuits and Signal Processing*, v. 103, n. 1, p. 173–186, 2020. 2, 16

SINGH, R.; BAGHEL, A.; AGARWAL, A. A review on vlsi floorplanning optimization using metaheuristic algorithms. *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, p. 4198–4202, 2016. 1, 16, 26

STEFANELLO, F. *Heuristic Approches for network problems*. Tese (Doutorado), 2015. UFRS. Orientação: Buriol, L. S.; Resende, M. G. C. 14

VALENTE, J. et al. Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs. *International Journal of Advanced Manufacturing Technology*, v. 54, n. 1-4, p. 251–265, 2011. 13

VENKATRAMAN, S.; SUNDHARARAJAN, M. A metaheuristic algorithm for vlsi floorplanning problem. *International Journal of Recent Technology and Engineering*, v. 8, n. 2 Special issue 5, p. 249–254, 2019. 1, 2, 16

WANG, Y.; CHEN, L. Two-dimensional residual-space-maximized packing. *Expert Systems with Applications*, Elsevier, v. 42, n. 7, p. 3297–3305, 2015. 1

WÄSCHER, G.; HAUβNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, v. 183, n. 3, p. 1109–1130, 2007. 6

WEI, L. et al. A skyline heuristic for the 2d rectangular packing and strip packing problems. *European Journal of Operational Research*, Elsevier, v. 215, n. 2, p. 337–346, 2011. 1

WEI, L.; ZHANG, D.; CHEN, Q. A least wasted first heuristic algorithm for the rectangular packing problem. *Computers & Operations Research*, Elsevier, v. 36, n. 5, p. 1608–1614, 2009. 1

WU, L. et al. A novel heuristic algorithm for two-dimensional rectangle packing area minimization problem with central rectangle. *Computers & Industrial Engineering*, Elsevier, v. 102, p. 208–218, 2016. 2, 16, 26

WU, Y.-L. et al. An effective quasi-human based heuristic for solving the rectangle packing problem. *European Journal of Operational Research*, Elsevier, v. 141, n. 2, p. 341–358, 2002. 1

YOUNG, F. et al. Handling soft modules in general nonslicing floorplan using lagrangian relaxation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 20, n. 5, p. 687–692, 2001. 14

ZHAN, Y.; FENG, Y.; SAPATNEKAR, S. A fixed-die floorplanning algorithm using an analytical approach. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, v. 2006, p. 771–776, 2006. 14

Apêndices

A Descrição de itens de cada instância

Instâncias do conjunto de referência arquivado no *Microelectronics Center of North Carolina (MCNC)* com origem no projeto de circuitos VLSI.

A.1 APTE

Item	Área	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
1	5744596	0,580	1,722	1825,34	3145,19	1826,47	3147,14
2	5744596	0,580	1,722	1825,34	3145,19	1826,47	3147,14
3	5744596	0,580	1,722	1825,34	3145,19	1826,47	3147,14
4	5744596	0,580	1,722	1825,34	3145,19	1826,47	3147,14
5	5836752	0,575	1,739	1831,97	3185,92	1832,04	3186,04
6	5836752	0,575	1,739	1831,97	3185,92	1832,04	3186,04
7	5836752	0,575	1,739	1831,97	3185,92	1832,04	3186,04
8	5836752	0,575	1,739	1831,97	3185,92	1832,04	3186,04
9	236236	0,346	2,888	285,90	825,98	286,01	826,29

Tabela 17 – Descrição de itens da instância APTE.

A.2 XEROX

Item	Área	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
1	797720	0,475	2,102	615,56	1294,92	616,04	1295,92
2	634550	0,378	2,642	489,75	1294,79	490,08	1295,65
3	3281530	0,511	1,956	1294,94	2533,51	1295,25	2534,12
4	3326855	0,504	1,983	1294,89	2568,49	1295,26	2569,22
5	635040	0,900	1,111	756,00	839,96	756,04	840,00
6	2253118	0,599	1,668	1161,73	1938,61	1162,23	1939,45
7	2012136	0,737	1,356	1217,76	1651,80	1218,14	1652,32
8	1160712	0,670	1,492	881,86	1315,97	882,02	1316,21
9	2737630	0,612	1,632	1294,38	2113,72	1295,17	2115,01
10	2511005	0,667	1,497	1294,16	1938,81	1295,13	1940,26

Tabela 18 – Descrição de itens da instância XEROX.

A.3 HP

Item	**Área**	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
1	478632	0,445	2,242	461,51	1035,90	462,04	1037,10
2	264600	0,540	1,851	378,00	699,84	378,09	700,00
3	205800	0,214	4,666	209,86	979,93	210,02	980,65
4	205800	0,214	4,666	209,86	979,93	210,02	980,65
5	205800	0,214	4,666	209,86	979,93	210,02	980,65
6	1803984	0,165	6,051	545,58	3303,92	546,01	3306,54
7	1803984	0,165	6,051	545,58	3303,92	546,01	3306,54
8	508032	0,125	8,000	252,00	2016,00	252,00	2016,00
9	1422960	0,150	6,666	462,00	3079,85	462,02	3080,00
10	508032	0,125	8,000	252,00	2016,00	252,00	2016,00
11	1422960	0,150	6,666	462,00	3079,85	462,02	3080,00

Tabela 19 – Descrição de itens da instância HP.

A.4 AMI33

Item	**Área**	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
1	44688	0,395	2,526	132,86	335,98	133,01	336,35
2	44982	0,314	3,176	118,85	377,97	119,01	378,49
3	22540	0,869	1,149	139,95	160,93	140,06	161,05
4	5831	0,411	2,428	48,95	118,99	49,01	119,11
5	20825	0,680	1,470	119,00	174,96	119,02	175,00
6	56840	2,900	0,344	406,00	139,83	406,49	140,00
7	69580	3,549	0,281	496,93	139,83	497,61	140,02
8	23324	0,607	1,647	118,99	196,00	119,00	196,02
9	34986	0,404	2,470	118,89	293,96	119,01	294,28
10	19159	0,739	1,352	118,99	160,94	119,04	161,01
11	31654	2,235	0,447	265,98	118,95	266,11	119,01
12	39984	2,823	0,354	335,97	118,97	336,08	119,01
13	14994	1,058	0,944	125,95	118,97	126,03	119,05
14	67522	0,490	2,038	181,89	370,96	182,02	371,21
15	36946	1,115	0,896	202,96	181,94	203,06	182,03
16	36946	1,115	0,896	202,96	181,94	203,06	182,03
17	9996	1,416	0,705	118,97	83,95	119,07	84,02
18	39102	2,210	0,452	293,96	132,94	294,12	133,02
19	63700	1,923	0,520	349,99	182,00	350,00	182,00
20	44100	0,444	2,250	139,93	315,00	140,00	315,16

Item	Área	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
21	41895	2,368	0,422	314,97	132,96	315,08	133,01
22	74480	0,237	4,210	132,86	559,96	133,01	560,59
23	18620	1,052	0,950	139,96	133,00	140,00	133,04
24	23275	0,759	1,315	132,91	174,95	133,04	175,12
25	30723	1,736	0,575	230,94	132,91	231,15	133,03
26	41895	2,368	0,422	314,97	132,96	315,08	133,01
27	17836	0,538	1,857	97,96	181,99	98,00	182,08
28	44100	1,000	1,000	210,00	210,00	210,00	210,00
29	47628	3,000	0,333	378,00	125,94	378,19	126,00
30	21658	0,653	1,529	118,92	181,98	119,02	182,12
31	14161	1,000	1,000	119,00	119,00	119,00	119,00
32	42483	0,333	3,000	118,94	357,00	119,00	357,18
33	9996	0,705	1,416	83,95	118,97	84,02	119,07

Tabela 20 – Descrição de itens da instância AMI33.

A.5 AMI49

Item	Área	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
1	5523672	0,528	1,893	1707,78	3233,62	1708,20	3234,42
2	1044288	0,432	2,312	671,66	1553,83	672,07	1554,78
3	2201472	0,461	2,166	1007,41	2183,66	1008,16	2185,28
4	4958800	0,522	1,913	1608,88	3079,96	1610,02	3082,14
5	737352	0,383	2,605	531,42	1385,93	532,03	1387,52
6	1642284	0,473	2,111	881,36	1861,95	882,02	1863,35
7	303800	0,403	2,479	349,90	867,82	350,07	868,24
8	802424	0,516	1,934	643,47	1245,75	644,13	1247,03
9	452760	0,471	2,121	461,79	979,95	462,02	980,45
10	181104	0,477	2,095	293,92	615,96	294,02	616,18
11	312228	0,457	2,185	377,74	825,96	378,02	826,57
12	323988	0,508	1,965	405,69	797,89	406,05	798,61
13	178752	0,395	2,526	265,72	671,96	266,02	672,71
14	375144	0,439	2,275	405,82	923,82	406,08	924,41
15	341040	0,483	2,068	405,86	839,80	406,09	840,29
16	256956	0,403	2,478	321,80	797,96	322,02	798,50
17	296352	0,518	1,928	391,80	755,89	392,06	756,38
18	422576	0,363	2,750	391,66	1078,00	392,00	1078,94

Item	**Área**	s_{min}	s_{max}	w_{min}	w_{max}	h_{min}	h_{max}
19	323792	0,474	2,107	391,76	825,97	392,01	826,50
20	201096	0,315	3,166	251,68	797,92	252,03	799,00
21	323792	0,474	2,107	391,76	825,97	392,01	826,50
22	346528	0,382	2,615	363,83	951,93	364,03	952,44
23	555660	0,432	2,314	489,94	1133,93	490,03	1134,13
24	212268	0,333	3,000	265,87	798,00	266,00	798,40
25	252448	0,41	2,434	321,72	783,87	322,05	784,68
26	254800	0,48	2,080	349,72	728,00	350,00	728,58
27	305760	0,433	2,307	363,86	839,87	364,05	840,32
28	551152	0,486	2,054	517,55	1063,99	518,01	1064,92
29	487060	0,492	2,028	489,52	993,86	490,07	994,97
30	947856	0,559	1,788	727,91	1301,83	728,09	1302,16
31	411600	0,373	2,678	391,82	1049,89	392,04	1050,47
32	296352	0,482	2,074	377,94	783,99	378,01	784,12
33	1799280	0,503	1,985	951,33	1889,86	952,07	1891,32
34	642880	0,487	2,049	559,54	1147,72	560,14	1148,95
35	334768	0,459	2,178	391,99	853,89	392,05	854,02
36	445312	0,45	2,218	447,65	993,83	448,08	994,78
37	682080	0,459	2,174	559,53	1217,72	560,13	1219,02
38	432768	0,463	2,156	447,63	965,94	448,03	966,80
39	285376	0,464	2,153	363,89	783,85	364,07	784,24
40	340256	0,451	2,214	391,73	867,94	392,03	868,59
41	310856	0,426	2,346	363,90	853,97	364,01	854,23
42	63504	0,444	2,250	167,92	378,00	168,00	378,19
43	484120	0,584	1,710	531,72	909,86	532,08	910,48
44	820260	0,483	2,066	629,43	1301,79	630,10	1303,17
45	780864	0,578	1,729	671,82	1161,94	672,03	1162,31
46	606816	0,418	2,388	503,64	1203,78	504,09	1204,87
47	352408	0,534	1,870	433,80	811,79	434,11	812,37
48	947856	0,559	1,788	727,91	1301,83	728,09	1302,16
49	290864	0,528	1,892	391,89	741,83	392,09	742,21

Tabela 21 – Descrição de itens da instância AMI49.