VINÍCIUS GANDRA MARTINS SANTOS

Orientador: Marco Antonio Moreira de Carvalho

BUSCA ADAPTATIVA EM GRANDES VIZINHANÇAS APLICADA À DETERMINAÇÃO DE LEIAUTES DE CIRCUITOS ELETRÔNICOS

Ouro Preto

Julho de 2017

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

BUSCA ADAPTATIVA EM GRANDES VIZINHANÇAS APLICADA À DETERMINAÇÃO DE LEIAUTES DE CIRCUITOS ELETRÔNICOS

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

VINÍCIUS GANDRA MARTINS SANTOS

Ouro Preto
Julho de 2017

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Busca Adaptativa em Grandes Vizinhanças Aplicada à Determinação de Leiautes de Circuitos Eletrônicos

VINÍCIUS GANDRA MARTINS SANTOS

Monografia defendida e aprovada pela banca examinadora constituída por:

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

Dra. Dayanne Gouveia Coelho
Universidade Federal de Ouro Preto

Dr. André Luis Silva
Universidade Federal de Ouro Preto

Ouro Preto, Julho de 2017

Resumo

Neste trabalho é proposto um novo algoritmo para solução do Problema de Determinação de Leiaute de Matrizes de Portas (*Gate Matrix Layout Problem*). Este problema combinatório é NP-Difícil e consiste em encontrar a disposição física das portas de um circuito eletrônico que minimize a quantidade de trilhas necessárias para a implementação do mesmo, diminuindo a área e o custo do circuito e melhorando seu desempenho. Métodos de solução deste problema possuem aplicabilidade prática na engenharia e indústria, por exemplo, na informática, no controle de processos, na automação de serviços bancários e comerciais e também na produção de bens de consumo. O problema abordado também pode ser utilizado para modelar com sucesso um grande conjunto de problemas de escalonamento em Pesquisa Operacional. Nesta monografia, é reportada a aplicação do método metaheurístico Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*) para solução do Problema de Determinação de Leiaute de Matrizes de Portas. Vale ressaltar que esta é a primeira aplicação reportada desta metaheurística à solução do problema abordado. Os experimentos computacionais envolvem 745 instâncias, entre elas instâncias reais e artificiais, de seis diferentes conjuntos disponíveis na literatura. Dentre todos os conjuntos de instâncias, o método divergiu em menos do que 1% dos valores das soluções ótimas, sendo que para 83,22% (ou seja, 620) das instâncias o valor ótimo foi encontrado com tempo de execução máximo de 10 minutos.

Abstract

In this work, we propose a new algorithm to solve the Gate Matrix Layout problem. This combinatorial problem is NP-Hard and consists of finding a physical arrangement of the gates of an electronic circuit that minimizes the number of necessary tracks in the printed circuit, thus, reducing the circuit area and the manufacturing costs, also improving its performance. Solution methods for this problem have direct practical application in engineering and industry, including, information technology, industrial processes control, banking and commercial automation and consumer goods production. The addressed problem can also be used to model a large set of scheduling problems in operations research. In this work, the application of the metaheuristic method Adaptive Large Neighborhood Search to provide solutions for the Gate Matrix Layout Problem is reported. It is noteworthy that this is the first reported application of the metaheuristic as a mean to the solution for this problem. The computational experiments considered 745 real-world and artificial instances, from six different benchmarks from the literature. Among all set of instances, the method solutions diverge from the optimal solutions by less than 1%, in average, and the optimal solution values were matched for 83,22% (i.e., 620) of the instances with a maximum running time of 10 minutes.

Dedico este trabalho em especial a minha mãe e a todos aqueles que estiveram ao meu lado durante essa jornada.

Agradecimentos

Agradeço aos meus pais por fazerem essa conquista possível e também pelo apoio, incentivo e suporte incondicional.

Agradeço a todos os docentes e servidores da Universidade Federal de Ouro Preto, lugar onde aprendi e amadureci ao longo desses anos.

Agradeço ao meu orientador, Marco Antonio, por todos os seus ensinamentos desde o meu ingresso na UFOP.

Agradeço a todos meus amigos pelo carinho, companheirismo e troca de conhecimento que compartilhamos ao longo dessa fase de nossas vidas.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Com a revolução tecnológica do século XX, a eletrônica vem cada dia mais investindo na miniaturização dos seus circuitos e componentes. Torres (2012) introduz os fundamentos de um circuito eletrônico, que pode ser definido como uma interconexão de elementos, ou componentes, ligados por fios condutores através dos quais a corrente elétrica pode fluir. Exemplos de componentes eletrônicos são transístores e diodos, dispositivos elétricos que transmitem corrente elétrica através de um condutor ou semicondutor. Estes componentes são dispostos sobre um substrato fino de material semicondutor, normalmente de silício, e são ligados através de fios, dando origem aos *circuitos integrados* ou *chips*. Diferentes disposições destes componentes geram diferentes leiautes para o mesmo circuito.

A miniaturização dá origem a microeletrônica, uma das áreas da eletrônica que permeia as mais diversas áreas. Ela está presente, por exemplo, na informática, no controle de processos industriais, na automação de serviços bancários e comerciais, e também na produção de bens de consumo, com aplicações práticas na engenharia e indústria. Além de oferecer todas as vantagens do pequeno tamanho, a miniaturização dos circuitos eletrônicos reduz o consumo de energia e diminui o tempo na comunicação entre os componentes do circuito, tornando-os mais baratos e rápidos.

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

Os circuitos podem ser representados por uma matriz de portas. Este dispositivo utiliza de duas dimensões de portas programáveis, uma dimensão AND e outra dimensão OR, em que as colunas representam as *portas*, cujas eventuais conexões são feitas por um *fio* e representadas

pelas linhas. Estes fios formam um conjunto de conexões entre portas que corresponde a uma *rede*. No circuito físico, as redes são implementadas por meio de *trilhas*. Diferentes trilhas devem ser utilizadas para alocar redes que utilizem uma mesma porta, no intuito de evitar sobreposições de redes. Deste modo, a matriz de portas é utilizada para elaborar circuitos lógicos combinatórios, compostos por n portas lógicas e m redes.

Na Figura 1.1(a) é apresentada uma matriz de portas contendo 6 portas enumeradas de P1 a P6 e 6 redes enumeradas de R1 a R6. As portas são representadas por linhas verticais enquanto as redes são representadas na horizontal por um conjunto de pontos conectados através de um fio. A rede R1 conecta as portas P3 e P4, a rede R2 conecta as portas P1, P2 e P6 e assim por diante. Note que o fio pode cruzar portas que não estão sendo conectadas, como pode ser observado na rede R2, em que o fio cruza as portas P3, P4 e P5, embora estas não façam parte da rede.

Na tentativa de otimizar o leiaute do circuito, deve-se permutar a ordem das portas da matriz, tendo em vista que na produção de circuitos eletrônicos esta operação não altera a lógica do circuito, desde que, os fios não se cruzem. O processo de permutação das portas consiste em alterar a ordem das portas na matriz com o objetivo de minimizar o número de trilhas, diminuindo consequentemente o tamanho do circuito. Na Figura 1.1(b) é apresentado um novo leiaute do circuito com permutação das portas P6, P5, P2, P1, P3 e P4. Observe que o novo leiaute pode ser compactado, ou seja, permite que diferentes redes sejam alocadas a uma mesma trilha. Este leiaute compactado, Figura 1.1(c), utiliza 3 trilhas, cada trilha com duas redes, {R5, R4}, {R2, R1} e {R3, R6}, enquanto o leiaute anterior, Figura 1.1(a), utiliza 6 trilhas, em que uma única rede é alocada por trilha.

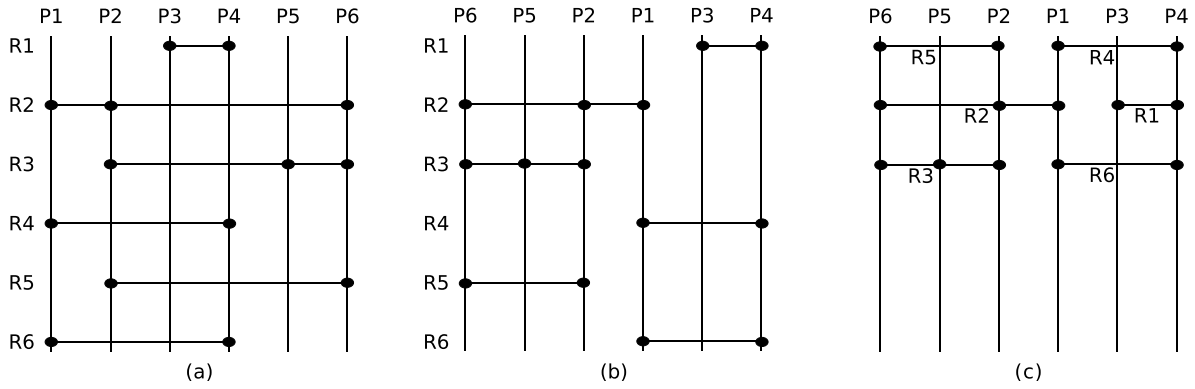

Figura 1.1: Matriz de portas (a) original, permutada (b), e compactada (c).

O Problema de Determinação de Leiaute de Matrizes de Portas (ou GLMP, do inglês *Gate Matrix Layout Problem*) consiste em determinar a permutação ótima de portas de modo a minimizar a quantidade de trilhas necessárias para implementar o circuito integrado correspondente e consequentemente minimizar a área e custo de produção do mesmo (Wing et al.,

1985).

Para a solução do GMLP, este trabalho propõe implementar o método Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS), uma metaheurística recente que utiliza buscas locais e perturbações para explorar uma porção ampla das possíveis soluções para problemas combinatórios. O ALNS foi escolhido para este trabalho pois são conhecidas diversas heurísticas de bom desempenho que podem ser incluídas no conjunto de heurísticas do ALNS. O ALNS é um algoritmo robusto que se adapta a várias características das instâncias e dificilmente fica preso em ótimos locais. Esta é a primeira aplicação do método ALNS ao GMLP reportada na literatura.

1.1 Justificativa

O GMLP foi provado ser um problema NP-difícil por uma redução ao problema de aumento de grafos de intervalo por Kashiwabara e Fujisawa (1979) (*apud* Linhares e Yanasse, 2002). Linhares e Yanasse (2002), demonstraram que o problema possui aplicações em diversas áreas, assim como diferentes problemas de formulação equivalente.

A aplicação prática deste problema tem ganhado destaque. Uma nova fase na história industrial do país foi inaugurada com a recente instalação, em Minas Gerais, da primeira fábrica de semicondutores do hemisfério sul, gerando equipamentos de alta tecnologia e maior valor agregado. Sendo assim, este trabalho pode contribuir para gerar conhecimento e inovação para este nicho industrial.

1.2 Objetivos

De maneira geral, o objetivo deste trabalho é elaborar uma heurística consistente para ser utilizada no problema de determinação de leiaute de matrizes de portas que permitam a obtenção rápida de soluções próximas da solução ótima. São objetivos específicos:

- Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre o GMLP;
- Realizar pesquisa para geração de embasamento teórico e revisão bibliográfica sobre o ALNS;
- Avaliar o método implementado considerando dados reais e também com problemas teste publicamente disponíveis, realizando uma análise crítica considerando outros métodos da literatura.

Além dos objetivos principais, outros produtos deste projeto de pesquisa serão trabalhos publicados em periódicos e eventos nacionais.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma: uma breve revisão da literatura será apresentada no Capítulo 2. A fundamentação teórica está descrita no Capítulo 3, seguida do desenvolvimento no Capítulo 4, em que o método proposto está detalhadamente ilustrado. O Capítulo 5 demostra os resultados obtidos pelo trabalho proposto considerando instâncias reais e artificiais da literatura. Por fim, a conclusão é apresentada no Capítulo 6.

Capítulo 2

Revisão da Literatura

Kashiwabara e Fujisawa (1979) (*apud* Linhares e Yanasse, 2002), utilizaram a redução do problema de aumento de grafos de intervalo para provar que o GMLP é NP-difícil. Mais tarde, Linhares e Yanasse (2002) demonstraram que o GMLP possui diferentes problemas de formulação equivalente: Problema de Corte Modificado (*Modified Cutwidth*), Problema de Minimização de Pilhas Abertas (*Minimization of Open Stacks Problem* – MOSP), Dobradura de Arranjos Lógicos Programáveis (*Programmable Logic Array Folding*, ou PLA Folding), *Interval Thickness*, *Node Search Game*, *Edge Search Game*, *Narrowness*, *Split Bandwidth*, *Graph Pathwidth*, *Edge Separation* e *Vertex Separation*.

O GMLP tem despertado o interesse de pesquisadores que vêm desenvolvendo métodos exatos, heurísticos e metaheurísticos desde a década de 80. Wing et al. (1985) representaram o problema por grafos de intervalo e propuseram um método heurístico para resolver o mesmo. Em seguida, Hwang et al. (1987) adaptaram o algoritmo *min-net-cut* ao problema e alcançaram resultados interessantes para a época. O mesmo algoritmo foi utilizado por Chen e Hou (1988), em que eles representaram as conexões do circuito de forma dinâmica e criaram um algoritmo $O(nlogn)$, em que n é o número de portas, que analisa a dualidade dos circuitos semicondutores de metal-óxido complementar (*Complementary Metal-Oxide-Semiconductor*, CMOS). Este algoritmo superou os resultados para todas as 5 instâncias utilizadas retiradas da literatura, devidas a Hwang et al. (1987).

Na década de 90, avanços foram alcançados utilizando inteligência artificial, métodos híbridos e metaheurísticas. Princípios de inteligência artificial foram utilizadas por Hu e Chen (1990) e Chen e Hu (1990), que apresentaram os algoritmos Gm-Plan e Gm-Learn respectivamente. O Gm-Plan utiliza planejamento hierárquico e políticas de meta-planejamento, ambas estratégias de inteligência artificial, enquanto o Gm-Learn se baseia nos resultados do Gm-Plan e adota uma métrica de aprendizado para modificar a heurística utilizada. Destaca-se o Gm-Learn, que foi testado utilizando dezesseis circuitos retirados da literatura e que, superou os melhores resultados para oito circuitos. Para os outros oito circuitos restantes, seis tiveram seus resultados igualados ao melhor resultado conhecido e em somente dois circuitos, piores

resultados foram encontrados.

Subsequentemente, Shahookar et al. (1994) apresentaram um método híbrido, *Genetic Beam Search*, implementando um algoritmo genético e um método *Beam Search*. Os melhores resultados obtidos foram em relação ao comprimento das redes. Ao final da década de 90, Linhares (1999) propôs uma estratégia de busca local, a Busca Predatória. Esta busca proposta determinou o melhor resultado até então para 4 instâncias e alcançou os mesmos resultados que Hu e Chen (1990) e Chen e Hu (1990) para todas as outras instâncias da literatura. Além disto, o número mínimo de trilhas foi alcançado em 12 das 25 instâncias utilizadas. No mesmo ano, Linhares et al. (1999) conseguiram superar os resultados do Gm-Plan e Gm-Learn em 3 circuitos, sendo superado em apenas 1 caso e tendo igualado todos os outros resultados previstos com a Otimização Microcanônica (MO), um procedimento derivado do *Simulated Annealing*.

Posteriormente, algoritmos memético e construtivos foram explorados por alguns autores. Mendes et al. (2002) e Mendes e Linhares (2004) propuseram algoritmos meméticos, com destaque para este último. Junto ao algoritmo memético eles propuseram uma busca local que restringe a área de busca e outro método que descarta todos os movimentos que não afetam o gargalo do problema, chamado de *colunas críticas*. Com isso foi possível alcançar todos os melhores resultados da literatura até o momento e apresentar um bom desempenho quando comparado com algoritmos anteriores, como Gm-Plan, Gm-Learn, *simulated annealing* e heurísticas construtivas. Outra abordagem foi o algoritmo genético construtivo (CGA) proposto por de Oliveira e Lorena (2002), que utiliza esquemas para construir soluções que otimizam o número de trilhas e o comprimento das redes. O CGA foi comparado com o MO (Linhares et al., 1999) e obteve os mesmos resultados (em número de trilhas) para todas as instâncias em um tempo consideravelmente menor, não sendo possível comparar o comprimento das redes por indisponibilidade de dados.

Apesar de algoritmos exatos não serem populares para o GMLP, De Giovanni et al. (2013) desenvolveram o algoritmo, *Branch-and-Cut* (B&C), exato para determinar o valor ótimo do problema de tempo de pilhas abertas (ou TOS, do inglês *Time of Open Stacks*), que é equivalente ao problema de determinar o menor comprimento das redes. O B&C foi executado por no máximo uma hora para cada uma das 330 instâncias utilizadas. B&C encontra o valor ótimo para todas as instâncias pequenas e médias até 49 portas. No mesmo trabalho, De Giovanni et al. (2013) também propuseram um algoritmo genético (GAG) para o problema de pilhas abertas (ou MOS, do inglês *Maximum Number of Open Stacks*) e o comparou com CGA de de Oliveira e Lorena (2002) e o *lower bound* gerado por B&C. O GAG, segundo os autores, demonstra ser mais confiável que o CGA, já que encontra os melhores resultados para o MOS e melhores médias de valores para o TOS, da mesma forma, o GAG encontra valores ótimos ou perto dos ótimos para o TOS ao ser comparado com o B&C.

Recentemente, heurísticas foram utilizadas por Santos e Carvalho (2015) e Gonçalves et al.

(2016). Santos e Carvalho (2015) propuseram uma heurística que se baseia na representação do problema por grafos e é composta por: dois métodos de pré-processamento, geração da solução inicial e aprimoramento por busca local. Os testes foram feitos utilizando 190 instâncias, entre elas instâncias inéditas no contexto do problema. A heurística foi capaz de igualar boa parte dos melhores resultados disponíveis, e melhorar alguns deles.

Gonçalves et al. (2016) propôs uma metaheurística para o MOSP composta por um algoritmo genético de chaves aleatórias viciadas, *Biased Radom-Key Genetic Problem* (BRKGA), e uma busca local. Para cada cromossomo o BRKGA aplica 4 fases: Representação e Decodificação do Cromossomo; Construção da Solução; Ajuste do Cromossomo e Avaliação. Na primeira fase, a solução é representada como direta, representação padrão do problema, e como indireta, vetor de chaves randômicas que é mais tarde decodificado. Seguindo para a segunda fase, a construção da solução é dividida em duas etapas: na primeira, o padrão a ser inserido na solução é escolhido, e na segunda, a posição na solução corrente é determinada para a inserção do padrão selecionado na etapa anterior. A segunda etapa da fase de construção da solução é definida por uma busca local que não respeita as regras impostas pelo cromossomo, sendo assim necessário a aplicação da fase de ajuste. A heurística ajusta a ordem dos genes de acordo com a posição de cada padrão na solução final. Além de melhorar a qualidade da solução, a fase de ajuste também reduz o número de gerações necessárias para obter a melhor solução. A fase final de avaliação leva em conta não só a quantidade de pilhas abertas para a função de *fitness* mas também o potencial de melhoria da solução, para que soluções com o mesmo valor de MOSP possam ser distinguidas. Atualmente, este método representa o estado da arte relacionado a metaheurísticas aplicada ao problema. Os testes aplicados ao BRKGA envolvem mais de 6000 instâncias existentes na literatura. As instâncias podem ser divididas como a seguir: Harvey; Simonis; Shaw; Miller e Wilson (Smith e Gent, 2005); Faggioli e Bentivoglio (Faggioli e Bentivoglio, 1998); VLSI (Hu e Chen, 1990); SCOOP[1]; Harder150 (De Giovanni et al., 2013) e Becceneri (Yanasse et al., 1999). Os quatro primeiros conjuntos de instâncias citados anteriormente foram selecionados para teste da solução proposta neste trabalho.

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

Capítulo 3

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica do Problema de Determinação de Leiaute de Matrizes de Portas e de métodos disponíveis na literatura que serão utilizados neste trabalho. Nas seções seguintes serão apresentados uma definição formal do problema GMLP, a definição do método de pré-processamento, a representação por grafos, um método de busca local, o sequenciamento de portas, e por fim a definição do método ALNS.

3.1 Problema de Determinação de Leiaute de Matrizes de Portas

De acordo com Linhares e Yanasse (2002), uma instância do GMLP pode ser representada pelo número m de redes, o número de portas n, e por uma matriz $m \times n$ $M = m_{ij} \rightarrow \{1, 0\}$, em que a entrada m_{ij} tem seu valor definido de acordo com a Equação 3.1.

$$m_{ij} = \begin{cases} 1 & \text{se somente se, a rede } i \text{ incluir a porta } j, \\ 0 & \text{caso contrário.} \end{cases} \tag{3.1}$$

A Tabela 3.1 apresenta a matriz M do circuito representado anteriormente na Figura 1.1(a), em que suas conexões são representadas por valores 1, e o restante das entradas possui valor 0.

Tabela 3.1: Exemplo de representação de um matriz binária.

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

Uma solução para o GMLP consiste em uma permutação $\pi = [1, ..., n]$ das portas da matriz M, em que $\pi(j)$ indica a posição da porta j em π. A matriz permutação M^π tem suas portas sequenciadas de acordo com a permutação em π e é possuidora da propriedade dos 1s consecutivos, de maneira que $m^\pi_{ij} = 1$ se, e somente se, o fio da rede i incluir ou cruzar a porta j. Desta forma, todos os elementos com valor 0 entre os elementos de valor 1 das extremidades de cada rede são considerados também tendo o valor 1. A propriedade dos 1s consecutivos evita que redes sejam sobrepostas, pois reserva espaço para os fios, o que é importante para o problema já que redes sobrepostas alteram a lógica do circuito. A Equação 3.2 descreve formalmente a definição das entradas da matriz M^π.

$$m^\pi_{ij} = \begin{cases} 1 & \text{se } \exists x, \exists y \mid \pi(x) \leqslant j \leqslant \pi(y) \text{ e } m_{ix} = m_{iy} = 1, \\ 0 & \text{caso contrário.} \end{cases} \tag{3.2}$$

Uma solução para o GLMP é avaliada de acordo com a soma de cada coluna da matriz M^π, em que a coluna de maior soma é chamada de *coluna crítica* ou *gargalo* e resulta no valor da solução. Essas colunas indicam a quantidade de redes que não podem ser alocadas na mesma trilha, pois são redes que utilizam a mesma porta simultaneamente e sobreposição de redes não é permitido. A Equação 3.3 ilustra formalmente como a solução é avaliada.

$$Z_{GMLP}(M^\pi) = \max_{j \in \{1,...,n\}} \sum_{i=1}^{m} m^\pi_{ij} \tag{3.3}$$

Na Tabela 3.2(a) e Tabela 3.2(b) duas soluções $\pi_1 = [6, 5, 2, 1, 3, 4]$ e $\pi_2 = [6, 1, 2, 5, 3, 4]$ são ilustradas contendo a propriedade dos 1s consecutivos que estão destacados em negrito. Observe que, a matriz compactada resultante da permutação de portas π_1 necessita apenas de 3 trilhas para ser implementada, tendo todas as suas colunas como críticas, enquanto a matriz resultante de π_2 necessita de 5, sendo P1 e P2 as colunas críticas.

Tabela 3.2: Matriz binária com adição dos 1s consecutivos em negrito e permutação π_1 (a) e π_2 (b).

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

(a)

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

(b)

Tendo em vista que o número possível de permutações é $n!$ e que diferentes permutações geram diferentes resultados, é de extrema necessidade determinar métodos computacionais eficientes que determinem permutações otimizadas, ou seja, circuitos mais compactados.

Dada uma matriz M, o GMLP tem como objetivo determinar uma matriz M^{π} correspondente tal que a maior soma dos elementos de qualquer coluna seja a menor possível, o que corresponde a minimizar a quantidade necessária de trilhas para implementar o circuito equivalente. Dito isto, a função objetivo do GMLP define uma permutação π, entre o conjunto de todas as permutações Π, de tal forma que a soma dos elementos em qualquer coluna seja minimizada. A função objetivo é representada na Equação 3.4.

$$\min_{\pi \in \Pi} Z_{GMLP}(M) \tag{3.4}$$

A forma de avaliar a solução dada pela Equação 3.3 não difere soluções como demonstradas na Tabela 3.3(a) e Tabela 3.3(b), em que as duas soluções π_1 e π_2 necessitam de 5 trilhas para serem implementadas. Apesar do mesmo valor final da solução, a Tabela 3.3(a) aparenta ser melhor por possuir menos 1s consecutivos.

Tabela 3.3: Matriz binária com adição dos 1s consecutivos em negrito e permutação π_1 (a) e π_2 (b).

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

(a)

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

(b)

Gonçalves et al. (2016) propuseram uma função de avaliação modificada para o MOSP, problema correlato ao GMLP, em que é possível diferir duas soluções semelhantes e identificar a solução com o maior potencial para posteriores melhoras. A Equação 3.5 demonstra formalmente esta função de avaliação (denominada aqui MGMLP), em que o valor da solução relaciona o valor da função objetivo $Z_{GMLP}(M)$ e o número médio de elementos não nulos por coluna da matriz M^{π}.

$$Z_{MGMLP}(M^{\pi}) = Z_{GMLP}(M) + \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} m_{ij}^{\pi}}{Z_{GMLP}(M) \times n} \tag{3.5}$$

A nova função permite diferenciar soluções semelhantes que, caso utilize-se a função original, seriam consideradas iguais. Aplicando esta nova função de avaliação os valores das soluções π_1 e π_2 são respectivamente

$$Z_{MGMLP}(M^{\pi_1}) = 5 + \frac{3+5+5+3+3+1}{5 \times 6} = 5,66$$

$$Z_{MGMLP}(M^{\pi_2}) = 5 + \frac{3+5+5+3+3+3}{5 \times 6} = 5,73$$

Após o cálculo da solução podemos afirmar que solução π_1 é melhor que π_2.

3.2 Pré-processamento de Portas Dominadas

O pré-processamento de portas dominadas é uma técnica que têm como objetivo eliminar a redundância nas instâncias do GMLP, tornando-as menores. Dados são caracterizados redundantes quando podem ser ignorados sem que haja perda na estrutura do problema.

De acordo com Yanasse e Senne (2010), uma porta P_i domina outra porta P_j qualquer, se e somente se, o conjunto de redes que utilizam a porta P_j for um subconjunto das redes que utilizam a porta P_i. Quando identificadas, portas dominadas podem ser removidas temporariamente da solução. Ao obter a solução final π para a instância reduzida, ou seja, sem portas dominadas, as mesmas devem ser sequenciadas em π imediatamente após suas respectivas portas dominantes. A Tabela 3.4 apresenta um exemplo no qual é possível reduzir a instância utilizando o pré-processamento de portas dominadas.

Tabela 3.4: Exemplo de uma instância (a) antes e depois (b) do pré-processamento de portas dominadas.

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

(a)

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

(b)

Analisando a Tabela 3.4(a), observa-se que a porta $P3$ é utilizada pelo conjunto de redes $\{R1\}$, enquanto a porta $P4$ é utilizada pelo conjunto de redes $\{R1, R4, R6\}$. De forma análoga, $P6$ e $P2$ são utilizadas pelo conjunto $\{R2, R3, R5\}$ e a porta $P5$ por $\{R3\}$. Com isso, podemos concluir que a porta $P4$ domina a porta $P3$ e a porta $P6$ domina $P2$ e $P5$. Sendo assim, a Tabela 3.4 (b) apresenta uma instância reduzida em que as portas $P3$ e $P4$, assim como as portas $P6$, $P5$ e $P2$, são consideradas como uma só porta, reduzindo a instância da Tabela 3.4 (a) de 6 para 3 portas. Em uma possível solução, a porta $P3$ será obrigatoriamente sequenciada após a porta $P4$, assim como as portas $P2$ e $P5$ serão sequenciadas após $P6$, sem que haja nenhum prejuízo a otimalidade da solução.

De acordo com Yanasse e Senne (2010), o pré-processamento de portas dominadas exige que o conjunto de redes seja comparado para cada par de portas. Consequentemente, teremos

uma complexidade delimitada por $O(n^2m)$ para este procedimento.

3.3 Representação por Grafos

Nesta seção é considerada a representação por grafos não direcionados apresentada por Yanasse (1997) para instâncias MOSP, um problema equivalente ao GMLP. Aplicada ao GMLP, a modelagem proposta define que os vértices correspondem às redes, que devem ser conectadas por uma aresta se, e somente se, possuírem pelo menos uma porta em comum.

Esta representação permite levar em conta a relação entre redes e portas e também das redes entre si. A Figura 3.1 apresenta a instância da Tabela 3.4 representada em forma de grafo. Note que as redes R2, R4 e R6 compartilhavam a porta P1, portanto existe uma aresta que liga as três redes entre si.

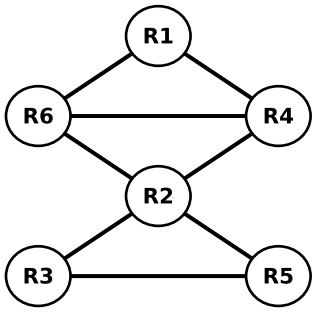

Figura 3.1: Exemplo representação de uma instância por grafo.

Note que, independente do pré-processamento de portas dominadas o grafo resultante é o mesmo. Isso quer dizer que tanto a instância representada na Tabela 3.4(a) quanto a instância reduzida representada na Tabela 3.4(b) resultarão no mesmo grafo, pois a junção de portas dominadas mantém a relação das portas em comum entre as redes.

3.4 Busca em Largura

A Busca em Largura (ou BFS, do inglês *Breadth-First Search*) é um algoritmo de busca em grafos que, dado um vértice inicial, explora sistematicamente todos os vértices de um grafo direcionado ou não-direcionado. A BFS explora inicialmente todos os vizinhos (vértices alcançáveis através de somente uma aresta) do vértice inicial, e à medida que são explorados os coloca em uma estrutura que segue a política de fila. Através desta operação, o algoritmo garante que nenhum vértice ou aresta seja explorado mais de uma vez. Ao final da exploração dos vértices vizinhos, a BFS remove o primeiro vértice da fila tornando-o o novo vértice inicial e repete-se o processo até que o alvo da busca seja encontrado ou todo o grafo tenha sido explorado.

Carvalho e Soma (2015) estabeleceram uma estratégia efetiva que diminui o número necessário de trilhas para soluções do GMLP. Os autores propõem sequenciar as portas levando em consideração o relacionamento entre as redes. A heurística proposta executa uma busca em largura no grafo e utiliza um critério guloso (vértice de menor grau) para guiar a BFS. Partindo deste princípio, a BFS é realizada no grafo no intuito de que as redes com menos portas sejam sequenciadas preferencialmente.

A Figura 3.2 demonstra como a BFS percorre o grafo apresentado na Figura 3.1, em que a expansão da vizinhança é representada por setas. O critério de escolha do vértice inicial, assim como dos demais vértices vizinhos a serem explorados, é o menor grau do vértice, em caso de empate, o critério passa a ser lexicográfico. A BFS seleciona inicialmente R1 como vértice inicial (em vermelho), e em seguida insere seus vizinhos R4 e R6 (em verde) na fila f, em que $f = \{R4, R6\}$. A primeira rede da fila é então retirada para ser explorada. Observe que o único vizinho não explorado de R4 é R2, atualizando a fila para $f = \{R6, R2\}$. A rede R6 por sua vez, não possui nenhum vizinho não explorado, então é somente retirado da fila e o próximo vértice a ter sua vizinhança analisada é o R2 (em amarelo). As redes R3 e R5 (em azul) são os únicos vizinhos não explorados de R2 e são inseridos na fila. Finalmente, os dois últimos vértices não possuem nenhum vizinho não visitado e são simplesmente removidos da fila e o processo termina. Ao final, a ordem λ de exploração dos vértices é obtida, sendo $\lambda = [R1, R4, R6, R2, R3, R5]$.

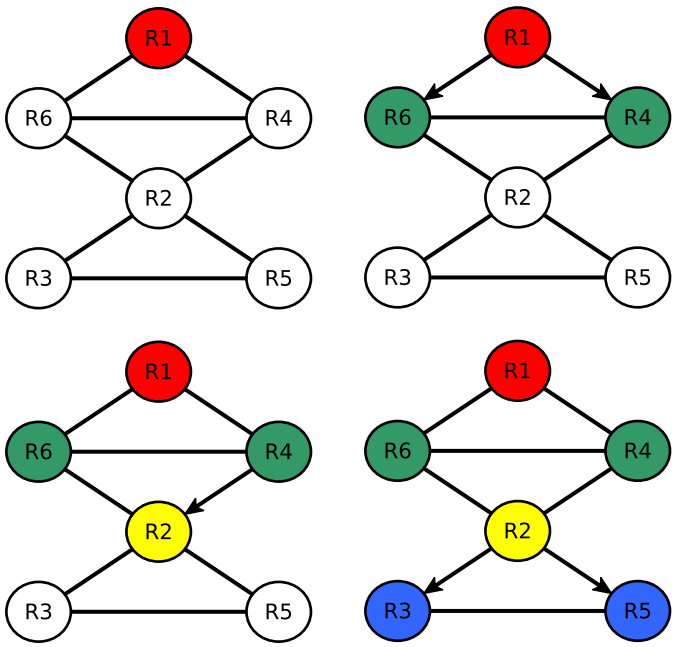

Figura 3.2: Exemplo de aplicação da BFS em um grafo. As setas representam a expansão da vizinhança.

3.5 Sequenciamento de Portas

Uma solução viável para o GMLP consiste em uma sequência π de portas, entretanto a BFS resulta em uma lista λ de redes. Com isso, é necessário obter uma permutação π de portas a partir da lista λ. Becceneri et al. (2004) propuseram um método para um problema equivalente que pode ser empregado no sequenciamento de portas do GMLP. O objetivo deste método é evitar que portas não relacionadas a determinadas redes sejam sequenciadas prematuramente, mantendo a relação estabelecida em λ e minimizando o número necessário de trilhas na solução.

O método proposto consiste em percorrer a lista λ analisando as portas utilizadas por cada rede, uma por vez. Quando todas as redes de uma determinada porta tiverem sido analisadas, esta é inserida ao final do vetor de solução π. Portas dominadas são inseridas imediatamente após suas portas dominantes, no intuito de obter uma solução completa ao final da execução do método.

Considere a instância de exemplo apresentada na Tabela 3.4(b) e a lista de redes $\lambda = [R1, R4, R6, R2, R3, R5]$ gerada pela BFS. A Tabela 3.5 apresenta a construção da solução π a partir de λ, em que cada linha representa as redes já analisadas e as portas sequenciadas.

Tabela 3.5: Exemplo sequenciamento de portas a partir da BFS.

Iteração	λ	π
1	R1	
2	R1, R4	
3	R1, R4, R6	P4, P3
4	R1, R4, R6, R2	P4, P3, P1
5	R1, R4, R6, R2, R3	P4, P3, P1
6	R1, R4, R6, R2, R3, R5	P4, P3, P1, P6, P2, P5

Inicialmente é observado que nenhuma porta possui exclusivamente um subconjunto das redes R1 e R4, portanto nas duas primeiras iterações não há sequenciamento de portas. Ao analisar a rede R6 observa-se que a porta P4 teve todas suas redes analisadas, então esta porta é inserida na solução seguida da porta dominada P3. A próxima rede a ser analisada é R2, verifica-se que a porta P1 é utilizada simultaneamente pelas redes R6, R4 e R2, todas analisadas anteriormente, então P1 é inserida na solução. Na iteração 5 a rede R3 é analisada mas nenhuma porta pode ser inserida na solução. Por fim, ao analisar a rede R5 na última iteração, a porta P6 e suas portas dominadas são incluídas na solução, considerando que as portas restantes são utilizadas simultaneamente pelas redes R2, R3 e R5. A Tabela 3.6 ilustra a matriz M^{π} após o sequenciamento de portas.

Observe que o resultado final foi de três trilhas, o mesmo resultado encontrado anteriormente no exemplo da Tabela 3.2(a). Este é o melhor resultado para a instância em questão, considerando que três redes compartilham a mesma porta simultaneamente e não podem di-

Tabela 3.6: Exemplo da solução após sequenciamento e adição dos 1s consecutivos em negrito.

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

vidir a mesma trilha em nenhuma circunstância.

3.6 Busca Adaptativa em Grandes Vizinhanças

Em otimização combinatória, técnicas de *busca local* são frequentemente utilizadas para encontrar bons resultados de forma rápida e pouco custosa. Este tipo de busca consiste em explorar o *espaço de busca*, definido por todas as soluções possíveis (conjunto Π), através de *vizinhanças*. Estas por sua vez, equivalem a um conjunto de soluções similares obtidas através de simples *movimentos* na solução original, como por exemplo trocar dois elementos de posição em um vetor para soluções que são representadas pelo mesmo. A busca local parte de uma solução inicial e é aplicada sucessivamente até que encontre um *ótimo local* ou sua condição de parada seja atendida. A busca atinge um ótimo local quando mudanças discretas não resultam em nenhum aprimoramento na solução, enquanto o *ótimo global* é o melhor valor possível para a função que define a qualidade das soluções, chamada de *função objetivo*. Diversas buscas locais podem ser utilizadas em simultâneo com o objetivo de escapar de ótimos locais e realizar uma busca robusta no espaço de busca de um problema. Essas buscas são frequentemente coordenadas por *metaheurísticas*, métodos de solução que utilizam estratégias de mais alto nível.

Ropke e Pisinger (2006) introduziram a metaheurística Busca Adaptativa em Grandes Vizinhanças (*Adaptive Large Neighborhood Search*, ALNS) tendo como base o método Busca em Grandes Vizinhanças (*Large Neighborhood Search*, LNS) proposto por Shaw (1998). O ALNS foi proposto para resolver o Problema de Roteamento de Veículos com Coleta e Entrega com Janelas de Tempo, em que sua solução é representada por uma permutação de elementos, assim como o GMLP.

O funcionamento do ALNS é representado através do fluxograma da Figura 3.3, em que a geração da solução inicial π_0 pode ser realizada através da utilização de heurísticas específicas para o problema tratado, de forma gulosa ou aleatória. Após a geração da solução inicial o ALNS inicia sua execução, primeiramente, selecionando uma vizinhança de remoção e outra de inserção através do método da roleta e em seguida aplica as vizinhanças na solução inicial para obter uma nova solução π'. Posteriormente, a solução gerada é analisada e pontos são

atribuídos as mesmas de acordo com seu desempenho, quanto melhor a solução melhor será a pontuação das vizinhanças. Caso seja o final de um segmento de execução, o ALNS suaviza os pontos da vizinhança e o algoritmo volta ao primeiro passo até que uma condição de parada seja satisfeita. Todos os métodos necessários para a execução do ALNS, serão descritos nesta seção.

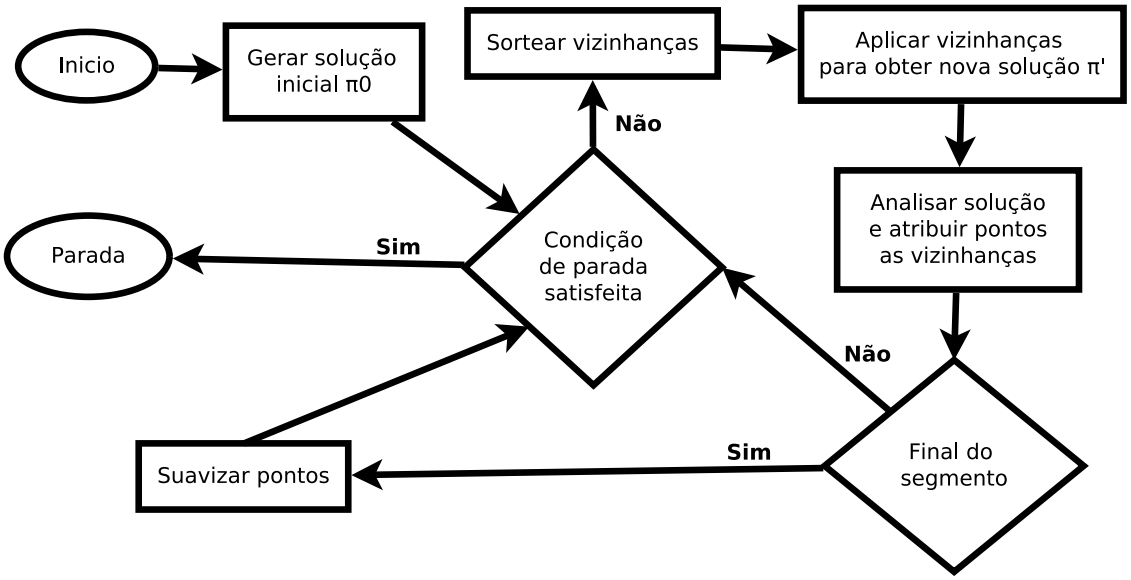

Figura 3.3: Fluxograma do funcionamento do ALNS.

Diferentemente do LNS, que utiliza procedimentos pesados como *Branch-and-Bound*, o ALNS utiliza um conjunto de diferentes heurísticas para buscas locais simples e rápidas que competem entre si para modificar a solução corrente explorando suas vizinhanças através de movimentos de remoção e inserção de elementos. A escolha dessas buscas locais influencia diretamente no resultado do ALNS. Inicialmente, deve-se escolher um conjunto N^- de heurísticas de remoção de elementos que intensificam e diversificam a busca, sendo que em cada iteração a heurística selecionada deve remover q elementos da solução. É necessário também escolher um conjunto N^+ de heurísticas de inserção de elementos que possa construir uma solução completa a partir de uma solução parcial. Essas heurísticas são geralmente baseadas em métodos gulosos de bom desempenho para o problema em questão.

Uma busca local é composta por uma heurística de remoção e uma heurística de inserção de elementos. O ALNS, por sua vez, possui diversas dessas heurísticas podendo assim combiná-las para criar variadas buscas locais que induzem diferentes vizinhanças, sendo assim, é de suma importância que a escolha de qual busca local utilizar seja feita de forma eficiente. O método de escolha de cada busca local é também um diferencial entre o ALNS e o LNS, tendo em vista que o LNS utiliza seus procedimentos de forma estática.

O ALNS utiliza um método estatístico baseado nas iterações anteriores do algoritmo para

decidir a prioridade da aplicação das heurísticas. A ideia é observar o desempenho de cada heurística ao longo das iterações e atribuir pontos para cada uma de acordo com o desempenho da mesma naquela iteração, tendo em vista que quanto mais a vizinhança $N_i \in \{N^+ \cup N^-\}$ contribui para uma solução, maior serão os pontos atribuídos à mesma e consequentemente, maior a chance de ser selecionada em iterações futuras. A Equação 3.6 apresenta como é calculada a probabilidade de uma vizinhança N_i ser selecionada entre ω vizinhanças, sendo os pontos da vizinhança i denotados por r_i.

$$P(N_i) = \frac{r_i}{\sum_{j=1}^{\omega} r_j} \tag{3.6}$$

O ALNS seleciona uma heurística de remoção e uma outra de inserção a cada iteração. A seleção é feita através do método da *roleta*. A roleta é representada por um intervalo $R = [0...1] \in \mathbb{R}$, em que cada vizinhança i recebe uma fatia da roleta proporcional à sua probabilidade $P(N_i)$ de ser escolhida. Sendo assim, quanto maior for a probabilidade da vizinhança ser escolhida, maior será a sua fatia de intervalo na roleta e consequentemente maior a chance da vizinhança ser selecionada. Na Figura 3.4, a vizinhança 4 tem 40% de probabilidade de ser escolhida, logo, possui a maior fatia na roleta, representada no intervalo entre $0,6$ e 1. Para que a seleção ocorra, um número $\nu \in (0, 1)$ é aleatoriamente escolhido e os valores das fatias são sequencialmente acumulados tal que a heurística correspondente à fatia cujo valor acumulado extrapolar o valor de ν é selecionada. Por exemplo, para $\nu = 0,73$ deve-se acumular o valor das quatro fatias da Figura 3.4 para que o valor de ν seja extrapolado. Considerando os valores acumulados das fatias como 0,12 (vizinhança 1), 0,37 (vizinhança 2), 0,60 (vizinhança 3) e 1 (vizinhança 4), esta última será selecionada.

Estes pontos utilizados para criação da roleta são ajustados de forma adaptativa. A execução do ALNS é dividida em blocos de iterações de tamanho θ chamados *segmentos*, durante os quais os pontos das heurísticas são cumulativamente calculados. Por exemplo, se a metaheurística for executada por 1000 iterações podemos ter 10 segmentos em que $\theta = 100$. Sendo $\overline{r}_{i,j}$ os *pontos observados* da heurística i no segmento j, podemos classificá-los em três categorias:

- σ_1, quando a última remoção ou inserção resultou no melhor resultado até o momento;
- σ_2, quando a última remoção ou inserção resultou em uma solução que não tinha sido encontrada anteriormente e cujo custo seja menor que o da solução corrente; e
- σ_3, quando a última remoção ou inserção resulta em uma solução viável não encontrada anteriormente, com o custo maior que o da solução corrente.

A terceira categoria tem em vista que soluções piores, quando viáveis, contribuem para uma perturbação na solução corrente e devem ser levadas em consideração. No final de cada segmento, os pontos acumulados para cada heurística passam por um processo de *suavização*,

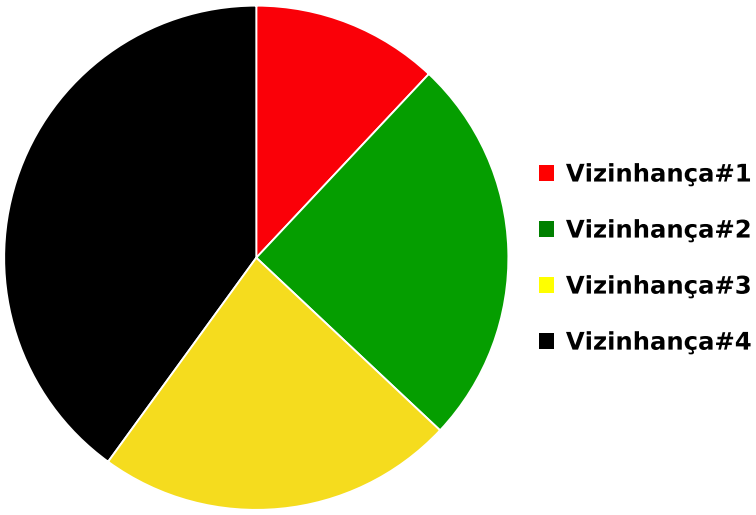

Figura 3.4: Exemplo de um roleta de seleção de vizinhança.

calculado conforme a Equação 3.7, em que a_i é o número de vezes que a heurística i foi chamada durante o segmento j. O fator de reação $\rho \in (0, 1)$ controla o quão rápido o ajuste de pontos do algoritmo reage às mudanças nos pontos de cada heurística. Quando $\rho = 1$ a roleta é baseada somente nos pontos do segmento imediatamente anterior, por outro lado, quando $\rho < 1$, os pontos dos segmentos anteriores são todos levados em consideração. Caso a heurística i não tenha sido chamada nenhuma vez durante o segmento, sua pontuação permanecerá a mesma do segmento imediatamente anterior. Esta estratégia está intimamente relacionada com os conceitos de memória de curto e longo prazo, comuns em otimização combinatória.

$$r_{i,j+1} = \begin{cases} r_{i,j} & \text{Se } a_i = 0. \\ \rho\frac{\overline{r}_{i,j}}{a_i} + (1-\rho)r_{i,j} & \text{Se } a_i \neq 0. \end{cases} \tag{3.7}$$

Pereira et al. (2015) apresentam uma versão mais simples para os pontos observados a cada iteração. Enquanto σ_1 consiste na mesma ideia do melhor resultado já encontrado, as outras duas categorias sofrem pequenas alterações:

- σ_2, quando a última remoção ou inserção resultou em uma solução cujo custo seja menor que o da solução corrente; e
- σ_3, quando a última remoção ou inserção resulta em uma solução que é aceita por um critério de aceitação, porém com o custo maior que o da solução corrente.

A vantagem destes novos cálculos é que desta forma não é necessário guardar a estrutura das soluções anteriores, facilitando assim a implementação do método, além de economizar

espaço e tempo de computação.

Cada nova solução gerada pelo ALNS com o custo maior que o da solução corrente passa por um critério de aceitação dinâmico dado por um método similar ao *simulated annealing*, que restringe gradativamente a qualidade das soluções. Desta forma, ao longo das iterações, as soluções para serem aceitas devem possuir valores cada vez melhores. Uma solução π' gerada a partir de outra solução π é aceita com probabilidade calculada de acordo com a Equação 3.8.

$$e^{-(f(\pi')-f(\pi))/T} \tag{3.8}$$

Originalmente, utiliza-se uma taxa de resfriamento exponencial, em que a temperatura T, sendo $T > 0$, decresce a cada iteração de acordo com a expressão $T = T \times c$, no qual $c \in \{0...1\}$ representa a taxa de resfriamento. A temperatura inicial é definida por $T_{start} = \psi \times f(\pi_0)/\ln 2$, em que o percentual de piora $\psi \in \{0,05...0,5\}$ determina que uma solução corrente entre 5% e 50% pior que a solução inicial π_0 é aceita com probabilidade de 50%, de acordo com o valor de ψ. O valor da taxa de resfriamento c neste trabalho é igual a $0,995$ e determinado de acordo com Pereira et al. (2015).

O ALNS é estruturado como apresentado no Algoritmo 1. O método parte de uma solução viável e a considera a melhor até então (linhas 1 e 2). Em seguida, um laço de repetição (linhas 3 a 23) é executado enquanto um determinado critério de parada não for atendido. Ao entrar no laço, uma heurística de remoção e outra de inserção são selecionadas através do método da roleta (linha 4), e em seguida aplicadas à solução atual π para gerar uma nova solução π' (linha 5). Posteriormente, o resultado de π' é comparado com a solução corrente π (linhas 6 a 13) e, caso seja melhor, se torna a nova solução corrente (linha 7). Uma segunda comparação é feita para checar se a solução corrente π é também melhor que a melhor solução π_{best} (linhas 9 a 12), se for melhor, π_{best} é atualizado (linha 10). No entanto, se π' possuir valor igual ou pior que a solução corrente, deve-se avaliá-la de acordo com o critério de aceitação (linhas 14 a 17) e, caso seja aceita, se torna a nova solução corrente (linha 15). Após a análise da nova solução, os pontos das vizinhanças são alterados (linha 18) conforme o valor recebido na linha 8, 11 ou 16. Ao final de cada segmento (linha 19), os pontos das vizinhanças são suavizados (linha 20) e a probabilidade nas roletas recalculada (linha 21). A cada iteração a última operação consiste em resfriar a temperatura T (linha 23) e por fim, quando o laço de repetição termina a melhor solução π_{best} é retornada (linha 25).

Algoritmo 1: Busca Adaptativa em Grandes Vizinhanças.

Input: solução viável π;
$\pi_{best} \leftarrow \pi$;
while *critério de parada não for atingido* **do**
    Escolha uma vizinhança de remoção N^- e uma vizinhança de inserção N^+ usando a roleta de seleção baseado nos pontos obtidos anteriormente $\{r_j\}$;
    Gere uma solução nova π' a partir de π usando a heurística de remoção e inserção escolhidas;
    if $f(\pi') < f(\pi)$ **then**
        $\pi \leftarrow \pi'$;
        $sigma \leftarrow \sigma_2$;
        if $f(\pi) < f(\pi_{best})$ **then**
            $\pi_{best} \leftarrow \pi$;
            $sigma \leftarrow \sigma_1$;
        end
    end
    else if π' *atender o critério de aceitação* **then**
        $\pi \leftarrow \pi'$;
        $sigma \leftarrow \sigma_3$;
    end
    Atualize os pontos r_j de N^- e N^+ conforme *sigma*;
    if *contador de interação for múltiplo de* θ **then**
        Suavize os pontos r_j acumulados;
        Atualize a probabilidade das vizinhanças na roleta conforme nova pontuação;
    end
    $T = T \times c$;
end
return π_{best};

O ALNS foi escolhido para este trabalho devido à suas diversas vantagens. Para a maioria dos problemas de otimização, incluindo o GMLP, diversas heurísticas de bom desempenho são conhecidas e podem ser incluídas no conjunto de heurísticas do ALNS. Devido ao tamanho das vizinhanças e também à variedade das mesmas, o ALNS é capaz de explorar grandes porções do espaço de busca de forma estruturada, resultando em um algoritmo robusto que se adapta a várias características das instâncias e dificilmente fica preso em ótimos locais. Sendo assim, o ALNS é indicado para problemas os quais vizinhanças pequenas não são suficiente para escapar de ótimos locais.

Capítulo 4

Desenvolvimento

Neste capítulo são descritas com detalhes as particularidades do ALNS implementadas nesse trabalho, incluindo critérios de parda do método e a implementação das vizinhanças utilizadas no ALNS para remoção e inserção de portas. As próximas seções descrevem nove métodos, cinco de remoção e quatro de inserção, que são combinados pelo ALNS para gerar diferentes buscas locais. Este capítulo também apresenta a implementação do método *2-swap*, introduzido no ALNS a fim de refinar a solução após a execução de uma vizinhança de remoção e outra de inserção, como método de busca local.

4.1 Solução Inicial

Como mencionado anteriormente, as heurísticas de inserção e remoção são aplicadas a partir de uma solução inicial. Neste trabalho a solução inicial foi gerada pelo conjunto dos métodos descritos nas Seções 3.2 a 3.5, proposto por Carvalho e Soma (2015).

O referido método consiste de uma heurística rápida e robusta projetada para um problema equivalente ao GMLP. É utilizado um método de pré-processamento dos dados de entrada, o problema é modelado utilizando grafos específicos e o relacionamento entre as redes é analisado para posteriormente obter-se uma sequência inicial de portas.

4.2 Critérios de Parada

O ALNS projetado para solução do GMLP atende a três critérios de paradas:

1. Temperatura T igual ou menor a $0,01$;
2. Solução igual a um limite inferior;
3. Limite de 100 iterações ininterruptas sem que haja melhora na solução.

O primeiro critério é adotado devido ao fato de que temperaturas T com valores muito baixos simplesmente não alteram a solução obtida pelo ALNS. O segundo critério é uma condição de otimalidade: caso o valor da solução iguale o maior número de redes em que uma mesma porta esteja presente, i.e., a maior soma de uma coluna desconsiderando-se os uns consecutivos, a solução é ótima e o ALNS pode ser interrompido.

O último critério é válido somente após o ALNS ter executado pelo menos metade da quantidade de iterações máximas definidas. Isso quer dizer que, se for definido 800 iterações como limite máximo, a condição de parada pode ser atingida no mínimo na iteração 500, sendo que, se uma melhora for encontrada na iteração 460, o contador é reiniciado e a nova iteração mínima será 560.

4.3 Vizinhanças de Remoção

Cada vizinhança de remoção recebe uma solução representada por uma sequência de portas. Essas vizinhanças ao serem executadas, removem portas da solução obtendo dois subconjuntos de portas: um representa a solução parcial e o outro as portas que devem ser inseridas na solução parcial por algum dos métodos de inserção. As portas que possuem a maior soma são chamadas de *colunas críticas* e serão importantes em algumas das vizinhanças. A Tabela 4.1 será utilizada como referência dos exemplos para algumas das seções seguintes.

Tabela 4.1: Exemplo de matriz não permutada com 1s consecutivos em negrito.

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

4.3.1 Remoção de Colunas Críticas (RCC)

A primeira vizinhança de remoção, proposta por de Carvalho e Soma (2016), calcula e então remove todas as colunas críticas da solução. É plausível acreditar em uma melhora da solução ao remover todas suas portas de maior valor para depois inserí-las em uma melhor posição. O exemplo da Tabela 4.1 tem $P2$ como coluna crítica, portanto, esta deve ser removida.

4.3.2 Remoção de Uns Consecutivos em Colunas Críticas (RFCC)

A segunda vizinhança de remoção, proposta por de Carvalho e Soma (2016), identifica as colunas críticas da solução e seus respectivos 1s consecutivos. Devem ser removidas portas que causem 1s consecutivos na coluna crítica, ou seja, portas que estão ativas nas extremidades

das redes em que se encontra um determinado 1 consecutivo da coluna crítica. Este método tem a intenção de diminuir a quantidade de 1s consecutivos nas colunas críticas para reduzir o valor da solução. O ALNS dispõe de duas implementações desta vizinhança, a primeira remove portas à esquerda da coluna crítica, enquanto a segunda remove as portas à direita.

Na Tabela 4.1, a coluna crítica é a $P2$, com 1 consecutivo em negrito. As portas $P1$, $P3$ e $P4$ compõe a rede $R1$, rede que cruza $P2$ causando 1 consecutivo. Sendo assim, as portas $P3$ e $P4$ serão removidas, tendo em vista que o método escolhido remove as portas à direita.

4.3.3 Remoção de Uns Consecutivos em Linhas (RFL)

A terceira vizinhança, proposta por de Carvalho e Soma (2016), é análoga a anterior, no entanto, a análise passa a ser por linhas. De forma aleatória, uma linha é selecionada e tem seus 1s consecutivos analisados. Então, as portas que causam os 1s consecutivos são removidas da solução. Na Tabela 4.2, se a linha $R2$ for sorteada a porta $P1$ é removida caso a vizinhança remova portas à esquerda do bloco de 1s consecutivos, e a porta $P4$ é removida caso a linha $R2$ ou $R3$ sejam selecionadas e a remoção seja pela direita.

Tabela 4.2: Exemplo da matriz não permutada com 1s consecutivos em negrito.

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

4.3.4 Remoção Aleatória (RRand)

A quarta vizinhança seleciona q portas de forma aleatória para serem removidas. Essa vizinhança tem o intuito de perturbar a solução, tendo em vista que utiliza um critério aleatório para a remoção. O valor de q é calculado conforme a Equação 4.1, proposta por Pereira et al. (2015).

$$q = \lfloor m - \sqrt{(1-u)(m-1)^2} + 0.5 \rfloor \tag{4.1}$$

A Equação 4.1 segue uma distribuição triangular e seleciona aleatoriamente um número entre $[1, m]$, em que m é o número de portas e u é uma variável aleatória entre $[0, 1]$. Essa operação não obtém um grande ganho na solução quando o número de portas é pequeno, mas pode ter grande impacto quando o número de portas é grande.

4.3.5 Remoção de Portas Relacionadas (RShaw)

A quinta vizinhança, proposta por Shaw (1998), remove portas que estão de alguma forma relacionadas. A similaridade neste caso é medida através da quantidade de 1s consecutivos. Uma porta é aleatoriamente selecionada, então as portas que possuem o mesmo número de 1s consecutivos que a porta já selecionada são também removidas. Com este método acredita-se que portas similares tem maiores chances de serem realocadas com sucesso. Na Tabela 4.3, se a porta P2, que possui um único 1 consecutivo (em negrito), for selecionada para ser removida, a porta P3 o será também por possuir o mesmo número de 1s consecutivos.

Tabela 4.3: Exemplo de matriz de solução com 1s consecutivos em negrito.

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

4.4 Vizinhanças de Inserção

Cada vizinhança de inserção recebe uma solução parcial e um conjunto de portas γ para ser inserido na solução. Cada porta do conjunto é selecionada aleatoriamente e inserida uma por vez na solução. Os métodos de inserção aceitam piora no valor da solução, sendo assim, mesmo que piore o valor de uma determinada solução uma porta sempre sera inserida na solução. Ao final, retorna-se como solução uma sequência de portas com o valor da função objetivo pior, igual ou melhor que a solução original, para que então o ALNS decida se a solução será aceita ou não.

4.4.1 Inserção Aleatória (IRand)

A primeira vizinhança de inserção insere cada porta em uma posição aleatória. Esta inserção em nenhum momento analisa a função objetivo, podendo gerar soluções consideravelmente melhores ou piores. A utilização desta inserção introduz diversidade à solução, que posteriormente pode ser aceita pelo ALNS mesmo que tenha um valor de solução pior.

4.4.2 Inserção Limitada por Coluna (ILC)

A segunda vizinhança, proposta por de Carvalho e Soma (2016), insere as portas na primeira posição que não piore a função objetivo, no entanto, o método recebe uma porta limite e uma direção para a inserção. Dado o método de remoção de 1s consecutivos em colunas críticas (Seção 4.1.2) e a Tabela 4.4 (a), temos a solução parcial $\pi = [1, 2]$, $P2$ como porta

limite e esquerda como direção. Deste modo, as portas $P3$ e $P4$ devem ser inseridas antes da porta $P2$. A Tabela 4.4 (b) demonstra um resultado possível após a inserção, em que a porta $P3$ foi inserida imediatamente antes da porta $P2$ e a porta $P4$ foi inserida logo após a porta $P1$, melhorando a função objetivo de 3 para 2 trilhas. Caso não seja possível inserir uma porta em uma posição que não piore a função objetivo, esta será inserida na posição que resulte no menor valor possível da função objetivo.

Tabela 4.4: Exemplo de matriz antes (a) e após (b) remoção e inserção de portas. 1s consecutivos em negrito.

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

(a)

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

(b)

4.4.3 Inserção na Melhor Posição (IBest)

A terceira vizinhança de inserção procura a melhor de todas as possíveis posições para inserir cada porta, levando em consideração o valor da função objetivo. Considere o método de remoção de 1s consecutivos em linhas (Seção 4.1.3) e a Tabela 4.5 (a), em que a porta $P4$ foi removida da solução. Na Tabela 4.5 (b) a porta removida é inserida na melhor posição possível e tem a função objetivo reduzida de 4 para 3 trilhas.

Tabela 4.5: Exemplo de matriz antes(a) e após (b) remoção e inserção, 1s consecutivos em negrito.

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

(a)

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

(b)

4.4.4 Inserção por Arrependimento (IReg)

A quarta e última vizinhança de inserção, Ropke e Pisinger (2006), tenta melhorar a heurística da melhor posição adicionando um mecanismo de *lookahead*. Ao selecionar de forma aleatória as portas para inserção, pode acontecer de deixar as portas mais difíceis de

serem inseridas por último, enquanto o ideal seria inserí-las o mais rápido possível. Desta forma, o método proposto não utiliza critério aleatório para selecionar as portas que devem ser inseridas, ao invés disso, as portas são ordenadas de acordo com o valor de *arrependimento* e então inseridas uma a uma na ordem estabelecida.

Para calcular o arrependimento deve-se primeiro encontrar a melhor e a segunda melhor posição de inserção de cada porta. Considere Δf_i^q como o custo da função objetivo ao inserir a porta i na q-ésima melhor posição. Por exemplo, Δf_i^2 indica o custo de inserir a porta i na segunda melhor posição. Desta forma, a cada iteração, a heurística de inserção escolhe a porta a ser inserida de acordo com a Equação 4.2, em que o arrependimento é a diferença no custo de inserir a porta na melhor posição possível ou na segunda melhor.

$$i = max_{i \in \gamma}(\Delta f_i^2 - \Delta f_i^1). \tag{4.2}$$

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

4.5 Busca Local

O clássico método de busca local *2-Swap* consiste em trocar dois elementos, ou portas, de posição. Inicialmente, todos os pares possíveis são gerados em ordem, o que consiste em um vetor de trocas com $n * (n-1)/2$ pares, e posteriormente o conteúdo do vetor é embaralhado. Por exemplo, considere uma instância com 4 portas, o vetor de trocas inicial será equivalente a [(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)], em que (1, 2) significa trocar a porta presente na posição 1 com a porta da posição 2. Em seguida, essas trocas são executadas uma a uma: a troca é mantida em caso de melhora no valor da solução, caso contrário, a troca é revertida. Desta forma, caracteriza-se uma busca local do tipo *first improvement*. Se uma troca gerar melhora na solução, o vetor de trocas é reembaralhado e as trocas começam a ser feitas novamente do começo do vetor. O método é executado até que todas as trocas sejam feitas sem que haja nenhuma melhora na solução.

A Figura 4.1 exemplifica a execução do método *2-swap* para uma solução que contém cinco portas na seguinte ordem [4, 1, 5, 3, 2]. Considere que o vetor embaralhado de trocas é [(2, 4), (1, 4), (2, 3), (1, 2), (3, 4), (1, 3)], desta forma a primeira troca selecionada será (2, 4). Na sequência, realiza-se a troca da porta na posição 2 com a porta na posição 4. Após a troca, o resultado final será [4, 3, 5, 1, 2], em que este resultado será mantido caso a função objetivo tenha melhorado, e será destrocado caso contrário. A próxima troca a ser realizada é (1, 4), considerando que a troca anterior foi mantida, o resultado final será [1, 3, 5, 4, 2], e assim por diante.

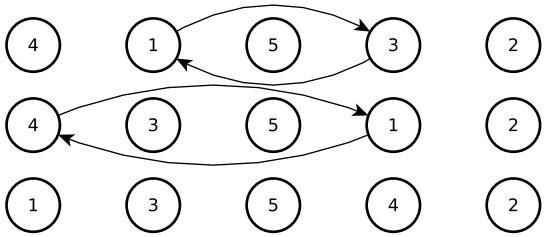

Figura 4.1: Exemplo do método 2-*swap*.

O método 2-*swap* foi utilizando no ALNS e é chamado uma vez a cada iteração entre as linhas *17* e *18* do Algoritmo 1 e somente modifica a solução em caso de melhora da função objetivo.

Capítulo 5

Experimentos Computacionais

Neste capítulo são apresentados os resultados dos experimentos computacionais realizados. Todos experimentos foram realizados em um computador com processador *Intel Core i7* de 3.6 GHz com 16 GB RAM, utilizando o sistema operacional Ubuntu 14.04 LTS. O método proposto foi codificado utilizando a linguagem C++ e compilado com gcc 4.8.4 e opções -O3 e *-march=native*.

5.1 Experimentos Preliminares

Como já visto no Capítulo 3, o ALNS necessita que sejam definidos alguns parâmetros para a execução do algoritmo. Para os experimentos apresentados neste capítulo, esses parâmetros foram determinados usando a ferramenta *irace* (López-Ibáñez et al., 2016), um pacote que pode ser encontrado na linguagem R. O *irace* é um método *offline* de configuração automática de algoritmos de otimização. Dado um conjunto de instâncias do problema, o método determina a combinação de valores mais apropriada para os parâmetros. A Tabela 5.1 demonstra os intervalos de valores que o irace considerou ao definir cada um dos parâmetros.

Tabela 5.1: Valores considerados pelo *irace* para cada parâmetro.

Parâmetro	Valores
σ_1	{5, 10, 15, 25, 30, 35, 40, 45, 50}
σ_2	{5, 10, 15, 25, 30, 35, 40, 45, 50}
σ_3	{5, 10, 15, 25, 30, 35, 40, 45, 50}
Percentual de piora ψ	[0,05 .. 0,50]
Fator de reação ρ	[0,00 .. 1,00]
Número de iterações	{300,400, 500, 600, 700, 800, 900, 1000}
Tamanho de segmento θ	{30, 40, 50, 60, 70, 80, 90, 100}
Número de iterações sem melhoras	{40, 60, 80, 100}

Os parâmetros definidos nos experimentos preliminares e seus respectivos valores são:

- $\sigma_1 = 15$;
- $\sigma_2 = 25$;
- $\sigma_3 = 5$;
- Percentual de piora $\psi = 0{,}41$;
- Fator de reação $\rho = 0{,}66$;
- Número de iterações = 800;
- Tamanho de cada segmento $\theta = 60$;
- Número de iterações sem melhoras = 100.

5.2 Comparação com Resultados da Literatura

Cinco conjuntos de instâncias foram utilizados e divididos em diferentes seções. Devido ao fator de aleatoriedade do método, foram realizadas 10 execuções independentes para cada uma das instâncias. O número de execuções foi escolhido com base na literatura. Em cada seção, uma breve apresentação do conjunto de instâncias é realizada, seguida de uma tabela contendo os resultados.

Cada tabela está organizada da seguinte maneira: a coluna *Instância* indica o nome de cada instância e as colunas seguintes m e n representam o número de redes e portas, respectivamente. Os resultados ótimos são apresentados na coluna OPT e o melhor resultado obtido pelo ALNS é representado na coluna S^*, sendo que os resultados obtidos que se equiparam ao valor ótimo conhecido são destacados em negrito. A média dos resultados encontrados nas 10 execuções do ALNS são representados na coluna S, a solução inicial é apresentada na coluna S_0 e o tempo de execução, expresso em segundos, se encontra na coluna T. Para melhor efeito de comparação de resultados, o *gap* (ou distância percentual) entre a solução ótima e a encontrada pelo ALNS é também apresentado na tabela e calculado conforme a Equação 5.1. Por fim, para que possamos checar a robustez do método, o desvio padrão entre as soluções encontradas após as 10 execuções para cada instância pode ser observado na coluna σ.

$$gap = 100 \times \frac{S^* - OPT}{OPT} \tag{5.1}$$

5.2.1 Instâncias Reais VLSI

O primeiro conjunto de instâncias foi introduzido por Hu e Chen (1990). Estas instâncias são reais e oriundas de empresas asiáticas. O conjunto possui 25 instâncias com matrizes de dimensões que variam entre 5×7 e 202×141. Os resultados são apresentados na Tabela 5.2.

Tabela 5.2: Instâncias reais VLSI. Valores em negrito indicam soluções ótimas.

Instância	m	n	OPT	S^*	S	S_0	T	gap	σ
v1	8	6	3	**3**	3,00	4	0,00	0,00	0,00
v4000	17	10	5	**5**	5,00	5	0,00	0,00	0,00
v4050	16	13	5	**5**	5,00	6	0,00	0,00	0,00
v4090	27	23	10	**10**	10,00	12	0,13	0,00	0,00
v4470	47	37	9	**9**	9,00	13	3,62	0,00	0,00
vc1	25	15	9	**9**	9,00	10	0,00	0,00	0,00
vw1	7	5	4	**4**	4,00	4	0,00	0,00	0,00
vw2	8	8	5	**5**	5,00	5	0,01	0,00	0,00
w1	21	18	4	**4**	4,00	5	0,00	0,00	0,00
w2	33	48	14	**14**	14,00	21	0,01	0,00	0,00
w3	70	84	18	**18**	18,00	34	8,96	0,00	0,00
w4	141	202	27	**27**	27,00	53	84,89	0,00	0,00
wan	7	9	6	**6**	6,00	6	0,00	0,00	0,00
wli	10	11	4	**4**	4,00	4	0,00	0,00	0,00
wsn	25	17	8	**8**	8,00	10	0,14	0,00	0,00
x0	48	40	11	**11**	11,00	18	2,60	0,00	0,00
x1	10	5	5	**5**	5,00	5	0,03	0,00	0,00
x2	15	6	6	**6**	6,00	6	0,11	0,00	0,00
x3	21	7	7	**7**	7,00	7	0,30	0,00	0,00
x4	8	12	2	**2**	2,00	2	0,00	0,00	0,00
x5	16	24	2	**2**	2,00	2	0,00	0,00	0,00
x6	32	49	2	**2**	2,00	2	0,00	0,00	0,00
x7	8	7	4	**4**	4,00	4	0,03	0,00	0,00
x8	16	11	4	**4**	4,00	4	0,21	0,00	0,00
x9	32	19	4	**4**	4,00	4	2,07	0,00	0,00
Média			7,12	7,12	7,12	9,84	4,12	0,00	0,00

Neste conjunto de instâncias, o ALNS obteve 100% das soluções ótimas. O tempo médio de execução é pouco maior que 4 segundos, no entanto, nota-se que essa média é influenciada fortemente pelas instâncias w3 e w4, que são as duas maiores instâncias deste conjunto. Todas as outras instâncias foram resolvidas em menos de 3,7 segundos. O *gap* total entre a solução inicial e a melhor solução é 38,20%, o que demonstra o importante papel que o ALNS desempenha neste trabalho. Além disso, o método foi capaz de encontrar todas as soluções ótimas em todas as 10 execuções independentes, resultando em um desvio padrão igual a zero e demonstrando robustez do método.

5.2.2 Instâncias SCOOP

O segundo conjunto possui instâncias reais para o problema de Minimização de Pilhas Abertas (um problema de formulação equivalente ao GMLP, conforme apresentado no Capítulo

2). Este conjunto, fornecido pelo SCOOP *Consortium*[1], contém 187 instâncias. No entanto, a maioria destas instâncias são muito pequenas (por exemplo, 2 linhas e colunas), portanto podem ser descartadas. Desta forma, o conjunto foi reduzido para 24 instâncias com dimensões significativas, variando de 10 linhas e 14 colunas a 49 linhas e 134 colunas. Os resultados são apresentados na Tabela 5.3.

Tabela 5.3: Instâncias reais SCOOP. Valores em negrito indicam soluções ótimas.

Instância	m	n	OPT	S^*	S	S_0	T	gap	σ
A_FA+AA-1	37	107	12	**12**	12,30	18	2,09	0,00	0,46
A_FA+AA-11	28	99	11	**11**	11,00	19	0,86	0,00	0,00
A_FA+AA-12	20	75	9	**9**	9,00	13	0,25	0,00	0,00
A_FA+AA-13	37	134	17	**17**	17,80	27	2,42	0,00	0,40
A_FA+AA-15	18	68	9	**9**	9,00	11	0,23	0,00	0,00
A_FA+AA-2	19	75	11	**11**	11,00	16	0,20	0,00	0,00
A_FA+AA-6	21	79	13	**13**	13,00	17	0,39	0,00	0,00
A_FA+AA-8	28	82	11	**11**	11,30	17	0,88	0,00	0,46
A_AP-9.d-3	16	20	6	**6**	6,00	6	0,08	0,00	0,00
A_AP-9.d-10	13	20	6	**6**	6,00	8	0,09	0,00	0,00
A_AP-9.d-11	21	27	6	**6**	6,00	7	0,14	0,00	0,00
A_AP-9.d-6	20	31	5	**5**	5,00	6	0,18	0,00	0,00
B_12F18-11	15	21	6	**6**	6,00	7	0,09	0,00	0,00
B_12M18-12	22	31	6	**6**	6,00	7	0,01	0,00	0,00
B_18AB1-32	11	15	6	**6**	6,00	7	0,03	0,00	0,00
B_18CR1-33	18	20	4	**4**	4,00	4	0,05	0,00	0,00
B_22X18-50	11	14	10	**10**	10,00	11	0,03	0,00	0,00
B_23B25-52	21	29	5	**5**	5,00	5	0,11	0,00	0,00
B_39Q18-82	10	14	5	**5**	5,00	6	0,00	0,00	0,00
B_42F22-93	10	18	5	**5**	5,00	5	0,02	0,00	0,00
B_CARLET-137	12	14	5	**5**	5,00	6	0,00	0,00	0,00
B_CUC28A-138	26	37	6	**6**	6,00	6	0,00	0,00	0,00
B_GTM18A-139	20	24	5	**5**	5,00	6	0,09	0,00	0,00
B_REVAL-145	49	60	7	**7**	7,00	9	1,71	0,00	0,00
Média			7,75	7,75	7,81	10,16	0,41	0,00	0,05

Neste conjunto de instâncias, o ALNS novamente encontrou 100% das soluções ótimas. O tempo de execução foi novamente baixo, média de 0,41 segundos, sendo que somente três instâncias demandaram maior esforço computacional, entre 1,71 e 2,42 segundos de execução do ALNS. As demais instâncias foram resolvidas em tempos de execução menores do que 1 segundo. Para este conjunto, a solução inicial possui um *gap* total de 30,85% em comparação com a melhor solução encontrada, demonstrando novamente a importância do ALNS na convergência das soluções. Somente em três das instâncias o método não conseguiu obter o valor ótimo em todas as 10 execuções, resultando em um desvio padrão igual a 0,05. O baixo

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

desvio padrão reforça a robustez do método neste segundo conjunto de instância.

5.2.3 Instâncias Faggioli e Bentivoglio

O terceiro conjunto de instâncias, proposto por Faggioli e Bentivoglio (1998), é composto por 300 instâncias artificiais para o MOSP. Os problemas foram agrupados em grupos de 10 instâncias cada de acordo com o número de linhas e colunas. A Tabela 5.4 apresenta os resultados obtidos, sendo que os resultados apresentados são a média das 10 instâncias de cada grupo.

Tabela 5.4: Instâncias agrupadas de Faggioli e Bentivoglio. Valores em negrito indicam soluções ótimas.

m	n	OPT	S^*	S	S_0	T	gap	σ
10	10	5,50	**5,50**	5,50	5,90	0,02	0,00	0,00
10	20	6,20	**6,20**	6,20	7,70	0,03	0,00	0,00
10	30	6,10	**6,10**	6,10	8,10	0,00	0,00	0,00
10	40	7,70	**7,70**	7,70	8,50	0,02	0,00	0,00
10	50	8,20	**8,20**	8,20	9,40	0,00	0,00	0,00
15	10	6,60	**6,60**	6,60	7,00	0,06	0,00	0,00
15	20	7,20	**7,20**	7,20	9,80	0,13	0,00	0,00
15	30	7,30	**7,30**	7,32	9,80	0,12	0,00	0,04
15	40	7,20	**7,20**	7,20	9,10	0,12	0,00	0,00
15	50	7,40	**7,40**	7,40	9,50	0,09	0,00	0,00
20	10	7,50	**7,50**	7,50	7,90	0,09	0,00	0,00
20	20	8,50	**8,50**	8,50	10,10	0,26	0,00	0,00
20	30	8,80	**8,80**	8,89	12,30	0,33	0,00	0,07
20	40	8,50	**8,50**	8,57	12,80	0,33	0,00	0,04
20	50	7,90	**7,90**	7,94	11,40	0,28	0,00	0,05
25	10	8,00	**8,00**	8,00	8,20	0,15	0,00	0,00
25	20	9,80	**9,80**	9,80	12,30	0,52	0,00	0,00
25	30	10,50	**10,50**	10,50	14,80	0,66	0,00	0,00
25	40	10,30	**10,30**	10,36	14,80	0,65	0,00	0,05
25	50	10,00	**10,00**	10,00	15,20	0,67	0,00	0,00
30	10	7,80	**7,80**	7,80	8,10	0,25	0,00	0,00
30	20	11,10	**11,10**	11,11	13,20	0,87	0,00	0,03
30	30	12,20	**12,20**	12,20	15,70	1,11	0,00	0,00
30	40	12,10	**12,10**	12,10	16,50	1,19	0,00	0,00
30	50	11,20	**11,20**	11,20	16,40	1,14	0,00	0,00
40	10	8,40	**8,40**	8,40	8,40	0,37	0,00	0,00
40	20	13,00	**13,00**	13,00	14,90	1,98	0,00	0,00
40	30	14,50	**14,50**	14,51	17,90	2,78	0,00	0,03
40	40	14,90	**14,90**	14,91	19,90	2,81	0,00	0,03
40	50	14,60	**14,60**	14,63	20,30	3,01	0,00	0,07
Média		9,30	9,30	9,31	11,86	0,67	0,00	0,01

Neste conjunto de instâncias, o ALNS obteve 100% de todas as soluções ótimas com um tempo médio de 0,67 segundos, sendo que somente três grupos de instâncias exigiram tempo médio de execução acima de 2 segundos. O *gap* total entre a solução inicial e a melhor solução encontrada foi de 27,49%, sendo que nenhuma solução inicial é equivalente a solução ótima. A solução média não se equipara a melhor solução em 9 dos 30 grupos de instâncias, gerando um *gap* total de 0,11 e um desvio padrão de 0,01%. O ALNS também demonstrou ser robusto para este conjunto de instâncias.

5.2.4 Instâncias do *First Constraint Modeling Challenge*

O quarto conjunto de instâncias foi proposto para o *First Constraint Modeling Challenge* (Smith e Gent, 2005) e contém 46 instâncias MOSP. A escolha deste subconjunto foi feita levando em conta as dimensões das instâncias e a ausência de estruturas especiais que podem facilitar a solução, tal como analisado por Yanasse e Senne (2010). A Tabela 5.5 apresenta os resultados e possui uma coluna i adicional para indicar a quantidade de instâncias em cada grupo. A média dos resultados é apresentada quando i for maior que 1.

Tabela 5.5: Instâncias selecionadas do *First Constraint Modeling Challenge*. Valores em negrito indicam soluções ótimas.

Instância	m	n	i	OPT	S^*	S	S_0	T	gap	σ
nwrsSmaller	20	10	2	3,50	**3,50**	3,50	4,00	0,00	0,00	0,00
nwrsSmaller	25	15	2	7,00	**7,00**	7,00	7,00	0,12	0,00	0,00
nrwsLarger	30	20	2	12,00	**12,00**	12,00	12,00	0,36	0,00	0,00
nrwsLarger	60	25	2	13,00	**13,00**	13,00	14,00	2,20	0,00	0,00
GP1	50	50	4	38,75	**38,75**	38,75	39,75	7,51	0,00	0,00
GP2	100	100	4	76,25	**76,25**	76,25	77,50	245,90	0,00	0,00
Shaw	20	20	25	13,68	**13,68**	13,68	14,60	0,29	0,00	0,00
Miller	40	20	1	13	**13**	13,00	13	2,61	0,00	0,00
SP4-1	25	25	1	9	**9**	9,00	11	0,16	0,00	0,00
SP4-2	50	50	1	19	**19**	19,30	27	4,06	0,00	0,46
SP4-3	75	75	1	34	**34**	34,00	40	22,17	0,00	0,00
SP4-4	100	100	1	53	**53**	53,00	63	62,69	0,00	0,00
Média				24,35	24,35	24,37	26,90	29,00	0,00	0,04

Para este conjunto foi obtida a solução ótima em 100% dos grupos de instâncias. O tempo de execução médio é de 29 segundos, no entanto, o mesmo é menor para a maioria dos conjuntos, sempre abaixo de 8 segundos. O tempo médio é fortemente influenciado principalmente pelo grupo de instâncias GP2 ($T = 245,90$ por instância) e pelas instâncias SP4-3 ($T = 22,17$) e SP4-4 ($T = 62,69$). O *gap* total entre a solução inicial e a melhor solução encontrada é de 10,35%, demonstrando também a qualidade da solução inicial gerada pelo método da literatura. O ALNS gerou soluções ótimas em todas as execuções independentes de maneira consistente, exceto para a instância SP4-2, o que gerou um desvio padrão de 0,46% para este

caso particular. A média do desvio padrão, no entanto, foi de 0,04%. De acordo com os dados reportados, conclui-se mais uma vez a respeito da robustez do ALNS implementado.

5.2.5 Instâncias Grandes MOSP

O quinto conjunto de instâncias, geradas aleatoriamente por Chu e Stuckey (2009), é composto por 200 instâncias MOSP de maiores dimensões. As dimensões variam entre 30 × 30 e 125 × 125 e foram agrupadas em grupos de 25 instâncias cada de acordo com o número de linhas e colunas. A Tabela 5.6 apresenta os resultados obtidos, sendo que os resultados apresentados são a média das 25 instâncias de cada grupo.

Tabela 5.6: Instâncias *Random* (Chu e Stuckey, 2009).

m	n	OPT	S^*	S	S_0	T	gap	σ
30	30	19,60	**19,60**	19,63	21,64	1,09	0,00	0,03
40	40	25,48	**25,48**	25,50	28,96	2,97	0,00	0,03
50	50	31,40	**31,40**	31,49	35,80	6,40	0,00	0,13
75	75	44,84	44,92	45,09	52,12	24,57	0,18	0,22
50	100	38,44	**38,44**	38,58	41,40	49,29	0,00	0,15
100	50	41,08	**41,08**	41,26	52,80	7,13	0,00	0,18
100	100	58,20	58,56	59,02	68,04	63,59	0,62	0,41
125	125	70,20	70,84	71,39	84,16	134,52	0,91	0,43
Média		41,15	41,29	41,50	48,11	36,20	0,21	0,20

Este conjunto de instâncias, com dimensões superiores aos anteriores, exigiu maior esforço do método proposto para sua solução. Com efeito, o ALNS não conseguiu determinar todas as soluções ótimas para todos os grupos de instâncias. Entretanto, foi possível determinar as soluções ótimas para 175 (87,5%) das 200 instâncias. O tempo médio de execução foi de 36,20 segundos, sendo que os grupos de dimensões 125 × 125, 100 × 100 e 50 × 100 foram os que mais exigiram tempo de execução.

O ALNS obteve um *gap* médio de 0,21%, o primeiro maior que zero até então, e *gap* total de 0,34%. Todavia, o *gap* médio foi 0,00% para 5 dos 8 grupos de instâncias. A diferença percentual entre a solução inicial e a melhor solução encontrada foi de 15,51%, justificando o uso do ALNS. O desvio padrão foi de 0,20%, porém, ainda assim o valor é baixo e demonstra robustez do método. A solução média está próxima da melhor solução encontrada pelo ALNS, com distância percentual de 0,51%, o que indica pouca divergência nos valores da solução ao longo das execuções.

5.2.6 Instâncias Larger & Harder

O sexto conjunto de instâncias, também geradas aleatoriamente por Carvalho e Soma (2015), é composto por 150 instâncias. Este conjunto é o que possui as maiores dimensões até então: 150 × 150, 175 × 175 e 200 × 200. Cada dimensão é composta por um grupo

de 50 instâncias, e cada grupo dividido em subgrupos de 10 instâncias contendo 2, 4, 6, 8 e 10 redes por porta, o que implica em diferentes densidades das matrizes geradas. A Tabela 5.7 apresenta os resultados obtidos, sendo que os resultados apresentados são a média das 10 instâncias de cada subgrupo.

Tabela 5.7: Instâncias Larger & Harder.

Instância	*OPT*	*S**	*S*	S_0	*T*	*gap*	σ
150-150-2	25,90	26,30	26,86	45,00	229,18	1,54	0,42
150-150-4	61,60	61,80	62,62	82,70	260,35	0,32	0,56
150-150-6	93,20	94,00	94,68	112,40	277,30	0,86	0,52
150-150-8	111,70	112,70	113,47	125,90	288,20	0,90	0,54
150-150-10	123,90	124,60	125,41	133,70	298,35	0,56	0,53
175-175-2	30,30	30,80	31,71	55,00	373,05	1,65	0,60
175-175-4	73,70	74,90	75,95	100,70	448,45	1,63	0,64
175-175-6	107,70	108,80	109,81	131,20	488,98	1,02	0,69
175-175-8	128,30	129,30	130,37	146,60	490,13	0,78	0,64
175-175-10	143,50	144,20	145,33	155,00	505,73	0,49	0,66
200-200-2	36,00	37,00	37,78	64,30	619,54	2,78	0,57
200-200-4	84,30	**84,20**	85,33	111,40	712,93	-0,12	0,74
200-200-6	121,50	121,80	122,98	146,10	779,63	0,25	0,76
200-200-8	147,10	148,30	149,51	167,60	788,67	0,82	0,73
200-200-10	162,80	164,70	165,72	177,70	837,26	1,17	0,71
Média	96,77	97,56	98,50	117,02	493,18	0,98	0,62

Neste último conjunto de instâncias, o método proposto conseguiu determinar as soluções ótimas para 43 (28,67%) das 150 instâncias, além de melhorar o resultado anterior presente na literatura para 7 (4,67%) instâncias e ser capaz de encontrar *gap* médio negativo para um dos grupos de instâncias. O *gap* médio foi pouco menor que 1% e o *gap* total igual a 0,81%. O *gap* total em relação a solução média e a melhor solução foi de 0,96%.

Por apresentar *gap* total de 19,95% em relação a solução inicial e a melhor solução, o ALNS se mostra também eficiente e necessário para este conjunto de instâncias. O tempo médio de execução foi de 493,18 segundos, sendo que as instâncias de dimensões 150 $\times$ 150, foram resolvidas em menos tempo, em torno de 250 segundos. As instâncias de dimensões 200 $\times$ 200 exigiram maior tempo de execução, em torno de 750 segundos. O desvio padrão encontrado foi o maior de todos os conjuntos de instâncias considerados nesse trabalho (0,61%), embora este seja um valor baixo, particularmente em se tratando de métodos heurísticos.

5.3 Análises Adicionais

No Capítulo 4, vizinhanças de inserção e remoção foram apresentadas e é de grande interesse analisar quais foram mais eficientes, para que em trabalhos futuros possa ser realizado um refinamento nesta parte importante do ALNS. Por exemplo, otimizar as vizinhanças com

uma alta pontuação ou retirar as com baixa pontuação. A análise de convergência da solução é outra análise importante a ser feita, por permitir concluir sobre o número de iterações exigidas pelo ALNS para gerar soluções de boa qualidade e também sua eficiência. Com base nessas necessidades, esta seção é dedicada a essas duas importantes análises. A Tabela 5.8 apresenta a média dos pontos de cada uma das vizinhanças para cada grupo de instâncias.

Tabela 5.8: Pontuação das heurísticas de remoção e inserção.

Conjunto	*RCC*	*RFCC*	*RFL*	*RRand*	*RShaw*	*IRand*	*ILC*	*IBest*	*IReg*
VLSI	31,65	37,58	37,20	32,07	37,36	57,70	21,86	77,68	71,56
SCOOP	57,99	58,24	57,79	54,68	57,51	84,55	34,55	133,08	115,50
F & B	64,05	60,03	61,95	57,47	65,01	99,81	29,77	141,84	129,29
Challenge	67,14	61,21	64,35	63,52	67,96	100,46	29,57	152,71	137,41
Chu Stuckey	70,20	63,23	69,59	65,61	80,10	115,24	31,04	157,76	146,46
Larger & Harder	74,31	62,18	69,60	61,88	84,74	117,01	30,05	159,15	148,24
Média	60,89	57,07	60,08	55,87	65,45	95,80	29,47	137,04	124,74

Ao analisar as vizinhanças de remoção, nota-se que os pontos estão bem distribuídos entre elas. Por exemplo, a Remoção de Portas Relacionadas (RShaw) obteve a maior pontuação (65,45), enquanto a Remoção Aleatória (RRand) obteve a menor pontuação (55,87), de forma que a distância percentual entre essas duas foi de 17,15%. No entanto, houve uma maior discrepância de pontos entre as vizinhanças de inserção. A Inserção na Melhor Posição (IBest) obteve a melhor pontuação (137,04), seguida da Inserção por Arrependimento (IReg), com uma distância de 8,97%. A Inserção Aleatória (IRand) também conseguiu uma quantidade considerável de pontos, com distância percentual de 43,05% da IBest. A Inserção Limitada por Coluna (ILC), contudo, obteve somente 29,47 pontos, o que representa uma distância percentual de 365,06% da IBest.

A alta discrepância entre ILC e as demais heurísticas provavelmente se dá pelo fato que a vizinhança ILC pode ser escolhida somente se as vizinhanças RFCC e RFL forem selecionadas primeiro, enquanto todas as outras vizinhanças de inserção podem ser chamadas a qualquer momento independente de qual vizinhança de remoção foi selecionada anteriormente.

A Figura 5.1 apresenta o gráfico *boxplot* com a análise de convergência do método. O eixo y representa o número das iterações nas quais o método obteve sua melhor solução para cada instância, e o eixo x representa cada um dos conjuntos de instâncias. Para cada conjunto, o valor aferido representa o valor da média das 10 execuções para cada instância. A linha vermelha corresponde a mediana, ou seja, metade dos valores estão abaixo desta linha, divididos em dois quartis, e a outra metade está acima desta linha, divididos em mais dois quartis. Os pontos vermelhos correspondem aos *outliers*, ou valores atípicos, valores que apresentam um grande afastamento dos demais.

Os conjuntos VLSI, SCOOP e *First Constraint Modeling Challenge* (denotado por C), possuem baixos valores de convergência, em que a mediana se encontra bem próximo de

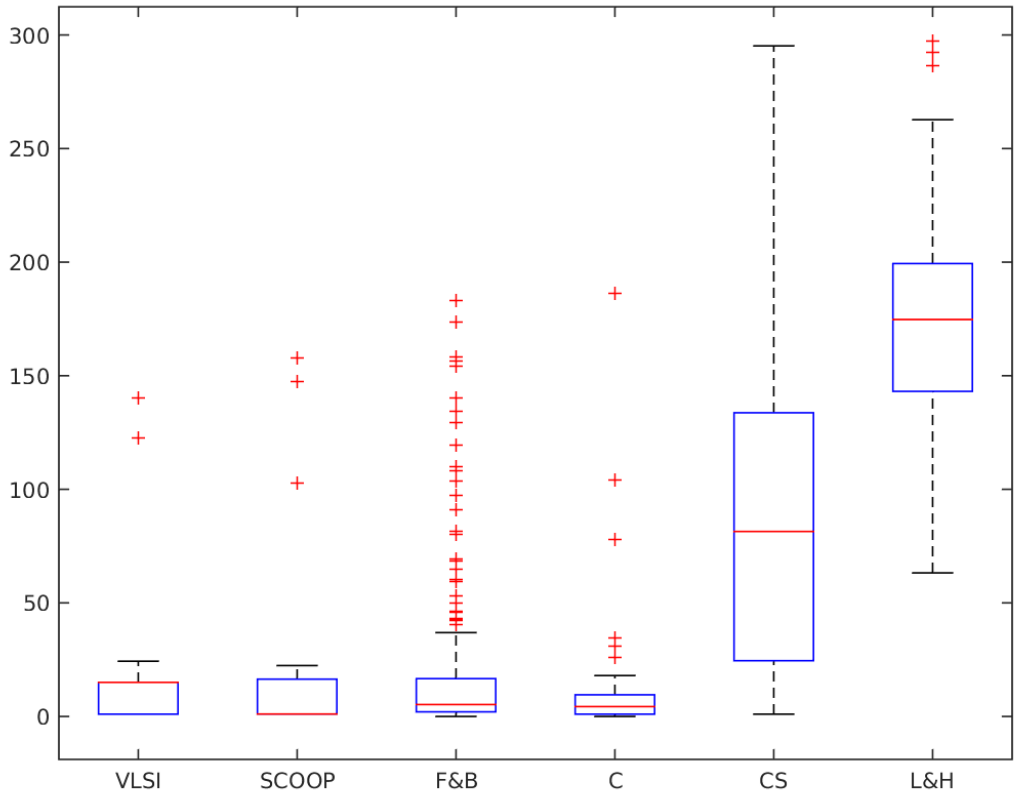

Figura 5.1: Análise de convergência do ALNS para os conjuntos de instâncias VLSI, SCOOP, *Faggioli e Bentivoglio* (F&B), *First Constraint Modeling Challenge* (C), *Chu e Stuckey* (CS) e *Larger & Harder* (L&H).

0 e seus valores máximos não estão muito distantes da mesma. Estes conjuntos também apresentam poucos *outliers* e possuem média aproximada de 24, 22 e 13 respectivamente. As instâncias de *Faggioli e Bentivoglio* (F&B) também possui o valor da mediana bem próximo de 0, e sua média é equivalente a 16,63. Os valores máximos deste conjunto também não estão muito distantes do valor da média. No entanto, por ser um conjunto com muitas instâncias (300), foi observado vários *outliers*, o que quer dizer que possui mais valores acima da linha da mediana do que abaixo. Ainda assim, nenhum dos seus valores foi superior a 200.

Os conjuntos *Chu e Stuckey* (CS) e *Larger & Harder* (L&H), por serem os dois conjuntos de maior dificuldade de solução e também com as maiores instâncias, apresentam um espalhamento maior dos seus valores. O primeiro obteve média igual a 86,14, um valor bem próximo de sua mediana, com valores mínimos e máximos que vai de 1 até quase 300. O conjunto *Chu e Stuckey* não apresenta *outliers*, o que significa que possui a mesma quantidade de valores abaixo e acima da mediana. Novamente, o último conjunto representado, possui sua mediana

bem próximo do valor da média (174,83) e apresenta somente 3 *outliers*. O conjunto apresenta valor mínimo de 63 iterações e máximo de 262.

O limite de iterações no ALNS implementado neste trabalho foi definido com valor 800, sendo que se a partir da iteração número 400 o método somente executa mais 100 vezes se não houver melhoria na solução. Com isso, podemos concluir que o método quase nunca executa mais de 500 iterações, mesmo nos últimos dois conjuntos de instâncias, em que o valor máximo para convergência, considerando os *outliers*, não excede 300 iterações. A média total considerando todos os conjuntos de instância gira em torno de 60 iterações.

Capítulo 6

Conclusões

O Problema de Determinação de Leiaute de Matrizes de Portas (ou GMLP, do inglês *Gate Matrix Layout Problem*) é um problema NP-Difícil com ampla aplicação industrial que visa, através de sequenciamento de portas, otimizar a área total de circuitos eletrônicos para reduzir o consumo de energia e diminuir o tempo de comunicação entre seus componentes do circuito, para que estes possam ser mais baratos, rápidos e compactos.

Neste trabalho foram apresentados a descrição formal do GMLP, seguido de uma revisão detalhada da literatura que considerou trabalhos que dizem respeito ao GMLP e problemas de formulação equivalente. A metaheurística Busca Adaptativa em Grandes Vizinhanças (ou ALNS, do inglês *Adaptive Large Neighborhood Search*) foi também detalhadamente estudada com o intuito de compor este trabalho. Com base nesse estudo foi possível compreender o problema e então projetar e implementar uma versão desta metaheurística. É importante ressaltar que esta é a primeira aplicação reportada do ALNS para solução do GMLP.

Os experimentos computacionais envolveram 745 instâncias de 6 diferentes conjuntos da literatura, incluindo instâncias reais e artificiais. Os resultados obtidos foram comparados com as soluções ótimas quando disponíveis e com as melhores soluções conhecidas nos outros casos. Os dados reportados revelam que aproximadamente 84% das soluções ótimas foram encontradas pelo método proposto neste trabalho. A maior divergência média dos valores obtidos e dos valores ótimos possui valor menor que 1%. Adicionalmente, o método proposto foi capaz de melhorar os melhores resultados conhecidos para sete das maiores instâncias consideradas. O tempo médio de execução foi baixo para todos os conjuntos de instâncias, sendo que para nenhuma instância este valor foi maior do que 10 minutos, um tempo de execução altamente desejável na prática.

Trabalhos futuros incluem a análise de novas heurísticas de inserção e remoção na ALNS, de maneira a aprimorar os resultados para o último conjunto de instâncias. Outro tema de interesse é a utilização de programação paralela, de maneira a permitir uma maior exploração do espaço de busca sem a implicação de aumento no tempo de execução.

Referências Bibliográficas

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

Carvalho, M. A. M. e Soma, N. Y. (2015). A breadth-first search applied to the minimization of the open stacks. *Journal of the Operational Research Society*, 66(6):936–946.

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

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

Chu, G. e Stuckey, P. J. (2009). Minimizing the maximum number of open stacks by customer search. In *International Conference on Principles and Practice of Constraint Programming*, pp. 242–257. Springer.

de Carvalho, M. A. M. e Soma, N. Y. (2016). Local search procedures for column permutation problems. Technical report, Universidade Federal de Ouro Preto, Departamento de Computação.

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

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

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

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

Hu, Y. H. e Chen, S.-J. (1990). Gm plan: a gate matrix layout algorithm based on artificial intelligence planning techniques. *IEEE transactions on computer-aided design of integrated circuits and systems*, 9(8):836–845.

Hwang, D. K.; Fuchs, W. K. e Kang, S. M. (1987). An efficient approach to gate matrix layout. *IEEE transactions on computer-aided design of integrated circuits and systems*, 6(5):802–809.

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

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

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

Linhares, A.; Yanasse, H. H. e Torreão, J. R. (1999). Linear gate assignment: a fast statistical mechanics approach. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 18(12):1750–1758.

López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L. P.; Birattari, M. e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

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

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

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

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

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

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

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pp. 417–431. Springer.

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

Torres, G. (2012). *Eletrônica para autodidatas, estudantes e técnicos*. Editora Novaterra.

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

Yanasse, H.; Becceneri, J. e Soma, N. (1999). Bounds for a problem of sequencing patterns. *Pesquisa Operacional*, 19(2):249–277.

Yanasse, H. H. (1997). On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, 100(3):454–463.

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