UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

TIAGO JOSÉ REIMANN

Roteamento Global de Circuitos VLSI

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Microeletrônica

Prof. Dr. Ricardo A. da Luz Reis
Orientador

Porto Alegre, abril de 2013

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Reimann, Tiago José

Roteamento Global de Circuitos VLSI / Tiago José Reimann. – Porto Alegre: PGMicro da UFRGS, 2013.

111 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2013. Orientador: Ricardo A. da Luz Reis.

1. Roteamento Global. 2. Síntese Física. 3. CAD. 4. VLSI. I. Reis, Ricardo A. da Luz. II. Título.

"Seja você quem for, seja qual for a posição social que você tenha na vida, a mais alta ou a mais baixa, tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá."

— AYRTON SENNA

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACE	*Adaptative Congestion Estimation*
AMMMR	*Adaptative Multi-source and Multi-sink Maze Routing*
ASIC	*Application Specific Integrated Circuit*
CFOMR	*Circular Fixed-Ordering Monotonic Routing*
CAD	*Computer Aided Design*
CMP	*Chemical Mechanical Polishing*
CI	Circuito Integrado
DLM	*Discrete Lagrange Multipliers*
FPGA	*Field Programmable Gate Array*
GRC	*Global Routing Cell*
ISPD	*ACM International Symposium on Physical Design*
ICCAD	*International Conference on Computer-Aided Design*
MSMMT	*Minimum Steiner min-max Tree*
MST	*Minimum Spanning Tree*
MWRST	*Minimum Weighted Rectilinear Steiner Tree*
SMMT	*Steiner min-max Tree*
RSMT	*Rectilinear Steiner Minimum Tree*
VLSI	*Very Large Scale Integration*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho apresenta a implementação de um roteador global de circuitos integrados capaz de tratar os problemas de roteamento atuais, utilizando como referência para avaliação os circuitos de *benchmark* publicados durante as competições de roteamento global realizadas no *ACM International Symposium on Physical Design* 2007 e 2008.

O roteador global desenvolvido utiliza como ferramenta principal a técnica de *rip-up and reroute* associada às técnicas de roteamento monotônico e *maze routing*, ambas com grande histórico de uso nas ferramentas acadêmicas descritas também neste trabalho. O desenvolvimento da ferramenta também possui características diferenciadas e únicas, com um novo método de ordenamento das redes durante a fase de *rip-up and reroute*.

Para a geração dos resultados foram definidas duas versões diferentes da ferramenta, sendo estas duas versões analisadas com duas diferentes técnicas de construção das árvores de roteamento, gerando no total quatro configurações da ferramenta. Como decisão de projeto, a versão principal utilizada no desenvolvimento e discussão dos resultados é a versão que prioriza a qualidade do roteamento, utilizando MSTs para construção das árvores de roteamento.

Os resultados mostram que o roteador global desenvolvido é capaz de gerar resultados com boa qualidade mesmo sem fazer uso de técnicas de identificação de áreas de congestionamento, sem otimizações pós-roteamento e sem nenhuma forma de ajuste (*tuning*) para os diferentes circuitos de *benchmark*, apesar de ainda ter tempo de execução acima dos apresentados por outras ferramentas acadêmicas.

O foco durante o processo de desenvolvimento e implementação da ferramenta foram os circuitos mais recentes, entretanto a ferramenta obteve ótimos resultados também para os circuitos publicados no ISPD 1998, gerando soluções com qualidade similar ou melhor que as reportadas na literatura.

A diferença dos resultados deste trabalho em relação aos melhores resultados dos roteadores globais com código disponível, para circuitos 3D lançados no ISPD 2008 é de, em média, 1,78%[1] na métrica de comprimento de fio sem considerar o custo das vias e de 15,56% considerando o custo da via como uma unidade de comprimento de fio (ISPD 2008), para a versão voltada a qualidade de roteamento. Já para a versão da ferramenta que busca a convergência o mais rápido possível a diferença foi de 3,39% e 16,32%, respectivamente. As maiores diferenças são encontradas nos circuitos mais difíceis de gerar uma solução sem violações. Isso mostra como as técnicas de identificação de região podem contribuir tanto para uma convergência mais rápida quanto para evitar que fios passem por rotas desnecessárias durante a fase de negociação.

Na métrica que avalia as vias como custo de uma unidade de comprimento, os resultados obtidos apresentam em média 18,67% maior comprimento de fio que os melhores

[1] Usando os valores da ferramenta desenvolvida como referência.

resultados da literatura, sendo que dois circuitos com solução sem violações[2] apresentam resultado com violações utilizando a ferramenta desenvolvida neste trabalho.

Palavras-chave: Roteamento Global, Síntese Física, CAD, VLSI.

[2]Na literatura alguns roteadores globais encontram solução sem violações para os circuitos bigblue2 e newblue1, entretanto os roteadores com código disponível, sem ajustes específicos para cada circuito, não foram capazes de gerar tal solução

Global Routing for VLSI Circuits

ABSTRACT

This work describes the implementation of an integrated circuit global router capable of handling the current routing problems, using as a reference the evaluation of benchmark circuits from the two global routing contests held in ISPD 2007 and 2008.

The developed global router uses rip-up and reroute as the main technique associated with monotonic and maze routing techniques, both with large history of use in academic tools, also described in this work. The tool also has distinctive and unique characteristics, with a new method of net ordering during the rip-up and reroute stage.

In order to generate the results were defined two different versions of the tool analyzed with two different techniques of routing tree construction, generating a total of four configurations. As a design decision, the major version used in the development and discussion of results is the version that prioritizes the routing quality, using MSTs for tree construction.

The results show that the global router developed is able to generate good results even without making use of techniques to identify congestion areas, without post-routing optimizations and without any form of tuning for the different benchmark circuits, despite having run time above other academic tools.

The focus during the development and implementation of the tool were the newer circuits, however the tool also obtained excellent results for the circuits released in ISPD 1998, generating solutions with similar quality or better than those reported in the literature.

The difference in the results of this work over the best results generated with the available code global routers for 3D circuits released in ISPD 2008 is, on average, 2.53% in wirelength metric without considering the cost of vias and 18.34% considering the cost of the vias as one wirelength unit (ISPD 2008), for the best routing quality version. As for the version of the tool that seeks convergence as soon as possible the difference was 3.82% and 17.03%, respectively. The largest differences were found in the most difficult circuits to generate a solution without violations. This shows how the techniques of congested region identification can contribute to both a faster convergence and to avoid unnecessary wire detours during the negotiation phase.

In the metric that evaluates the cost of vias as one wirelength unit, the results show an average of 22.5% greater wirelength than the best results found in literature. Also, the developed global router was unable to find a violation free solution for two circuits that are known to have a violation free solution[3].

Keywords: Global Routing, Physical Synthesis, CAD,VLSI.

[3]In the literature some global routers are able to find a solution without violations for newblue1 and bigblue2 circuits, however routers with available code, without tuning for each circuit, were not able to generate such a solution.

1 INTRODUÇÃO

O problema de roteamento existe desde os primórdios da eletrônica, inicialmente aplicado a placas de circuito impresso. Com o avanço da tecnologia e o surgimento da Microeletrônica, este problema passou a outras dimensões, ganhando progressivamente maior importância dentro do fluxo de projeto de circuitos VLSI.

Tal evolução abriu espaço para um grande aumento do número de publicações sobre o tema, gerando uma enorme gama de trabalhos abrangendo diversas abordagens do problema não só do ponto de vista da resolução do problema mas também a forma de implementação e técnicas de programação utilizadas.

Por se tratar de um problema de otimização combinatória, sabe-se que o problema de roteamento é NP-Completo (SZYMANSKI, 1985). Mesmo problemas de roteamento mais restritos, como roteamento de canal e roteamento com apenas uma camada, são problemas NP-Completo (SZYMANSKI, 1985; RICHARDS, 1984). Não só esse fato, mas também por atualmente circuitos VLSI alcançarem a exorbitante quantidade de bilhões de transistores, este problema é de difícil resolução computacional. Estes fatores tornam a abordagem em uma única fase computacionalmente inviável.

Assim, a abordagem tradicional de roteamento divide o problema em duas fases. A primeira, chamada de roteamento global, define as regiões pelas quais os fios irão passar, baseada na capacidade de roteamento destas regiões e da necessidade de alocação de recursos das redes que passam pelas fronteiras destas regiões, ou seja, não há conhecimento sobre as rotas internas destas regiões. A definição real das coordenadas pelas quais passarão as linhas de metal (redes) é feita na segunda fase, chamada de roteamento detalhado, como exemplificado na Figura 1.1. As diferenças entre essas duas etapas serão abordadas a seguir.

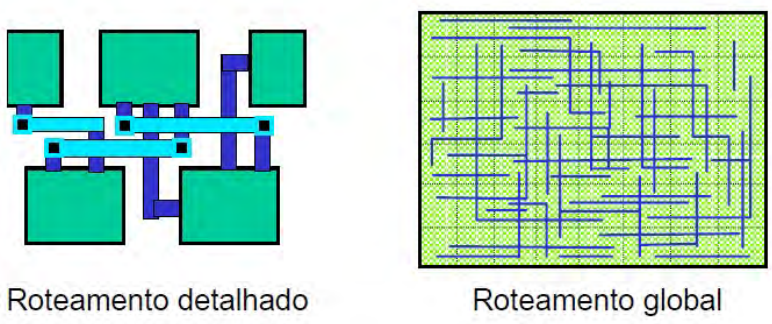

Figura 1.1: Exemplificação da diferença entre roteamento detalhado e global. Adaptada de (JOHANN, 2001)

Dentro do contexto de roteamento global, recentemente ocorreram duas competições (*ACM ISPD Routing Contests 2007-2008*) que provocaram uma grande evolução das ferramentas acadêmicas da área e serviram como novo estímulo para novas publicações, disponibilizando vários circuitos de teste (*benchmarks*) e também o código-fonte dos roteadores participantes das competições. Isso propiciou que novos trabalhos tivessem uma base de comparação de altíssimo nível e circuitos de teste com as características de tecnologias atuais usadas na fabricação de circuitos integrados.

Mesmo após tal evolução na área, ainda existe muito espaço para pesquisa em técnicas e fluxos de abordagem para resolução do problema. Um dos tópicos com maior foco é o roteamento dirigido a desempenho, onde o atraso das redes é considerado na construção das árvores de roteamento, dando privilégio para caminhos críticos.

Outro fator importante e perceptível em trabalhos mais recentes é a expansão do problema para englobar outras questões que são importantes em circuitos integrados atuais, como temperatura do circuito e número de vias.

Este trabalho está organizado da seguinte forma: no capítulo 2 são apresentadas brevemente algumas técnicas usadas em roteamento de redes e os métodos de tratamento do problema de roteamento global mais conhecidos. No capítulo 3 são mostrados os principais trabalhos apresentados nos dois concursos sobre roteamento global e os trabalhos recentes que delimitam o estado da arte atual. No capítulo 4 é apresentado o roteador global desenvolvido neste trabalho e os resultados deste aplicado aos circuitos de teste. Por fim, seguem as conclusões.

1.1 Roteamento Global

O roteamento global, como citado anteriormente, faz a definição das regiões por onde passarão as conexões do circuito. Tais regiões são definidas para serem suficientemente pequenas para que sejam tratáveis posteriormente pelo roteamento detalhado. O roteador global trata o CI por completo, diferente do que acontece com o roteador detalhado, que pode tratar apenas uma das regiões por vez.

Em geral, o roteamento global pode ser dividido em três etapas: definição das regiões, assinalamento das regiões e assinalamento dos pinos de cruzamento (*crosspoint assignment*). A definição das regiões, por ser feita com base nos parâmetros tecnológicos, é tida como parâmetro de entrada em ferramentas acadêmicas, sendo disponibilizada juntamente com o circuito de teste. O esforço maior de um roteador global está na fase de assinalamento de regiões. Nesta fase é identificada a sequência de regiões por onde passará cada rede, sendo o congestionamento de cada região o foco e podendo ser considerada também a necessidade de temporização da(s) rede(s) e outros fatores. O principal objetivo de um roteador global é evitar o excesso de congestionamento, para que o roteador detalhado possa concluir o roteamento. O assinalamento de pinos é a fase onde são definidos os pontos nas fronteiras onde passará cada rede que entra ou sai de uma determinada região. Isso permite que o roteamento detalhado considere cada região independentemente. Da mesma forma que a definição das regiões, nem sempre o assinalamento de pinos é feito dentro do roteador global propriamente dito.

Assim podemos definir um problema de roteamento global como descrito a seguir. Seja dado um grafo *G* especificando um conjunto de vértices *V* e um conjunto de arestas *E*, como ilustrado na Figura 1.2. Assim, cada vértice $v_i \in V$ corresponde a uma determinada região (ou célula) 3D na grade de roteamento global, e cada aresta $e_{ij} \in E$ corresponde a uma fronteira entre vértices adjacentes, contendo uma determinada capa-

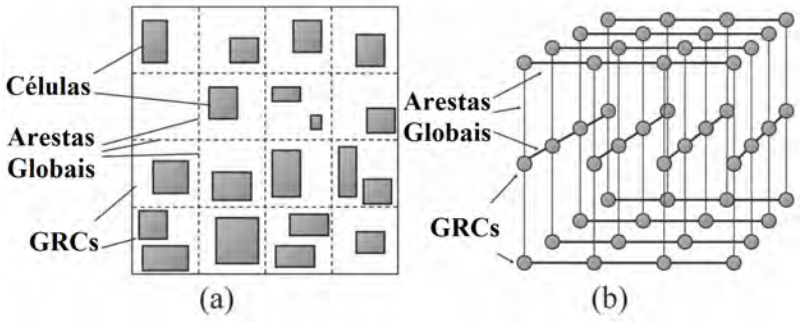

Figura 1.2: Decomposição em células globais e grafo correspondente. (MOFFITT, 2008a)

cidade de alocação. Esta região é dita 3D por existir mais de uma camada de metal para roteamento, formando um grafo 3D, conforme visto na Figura 1.2. Também há um conjunto de redes *N*, onde cada rede $n_i \in N$ é composta de um conjunto de pinos P_i, com cada pino correspondendo a um vértice v_i.

As ferramentas de roteamento global não necessariamente fazem uso de um grafo 3D para a resolução do problema, ao invés disso, fazem uso de um mapeamento de capacidades antes da execução do roteamento (transformando o problema 3D em 2D) e posterior assinalamento de camadas. Esses detalhes serão tratados mais adiante neste trabalho.

1.2 Roteamento Detalhado

O roteamento detalhado tem por função definir exatamente o caminho de todas as conexões do circuito. Isso inclui a posição de cada rota, os materiais, os furos (e suas posições) e as dimensões físicas do material usado (JOHANN, 2001).

Como dito anteriormente, por se tratar de um problema muito complexo para ser aplicado sobre um circuito completo, o roteamento detalhado é aplicado a regiões suficientemente pequenas, definidas na etapa de roteamento global (ou anteriormente, após o posicionamento das células no circuito). Assim, o roteamento detalhado é normalmente resolvido roteando-se uma região por vez, em uma ordem predefinida. Essa ordem é determinada por vários fatores, incluindo a criticalidade de roteamento de certas redes e o número total de redes passando pela região.

As características do problema de roteamento dependem amplamente da topologia da região de roteamento, podendo consistir de uma ou mais camadas, podendo camadas adjacentes serem interligadas através de furos na camada de isolamento, conhecidos como vias. Na maioria do modelos que consideram múltiplas camadas, essas camadas possuem direções restritas, podendo ter apenas segmentos de fio verticais ou horizontais.

Diferentes estratégias foram (e tem sido) desenvolvidas com uma grande variedade de objetivos, mas todos problemas de roteamento detalhado compartilham algumas características em comum, principalmente no que diz respeito às restrições geométricas impostas pela tecnologia alvo e, obviamente, ao fato de dois fios de redes diferentes não poderem cruzar-se na mesma camada.

Os objetivos primários do roteamento detalhado são completar as rotas para todas redes e obedecer as restrições de temporização do circuito. Vários objetivos secundários também são considerados, como: minimização do número de vias e desvios, minimizar *crosstalk* e atraso em redes críticas, minimizar comprimento médio ou total das redes, entre outros.

Para abordar este problema existem vários modelos propostos na literatura, que fogem ao escopo deste trabalho.

2 ROTEAMENTO GLOBAL DE CIRCUITOS INTEGRADOS

Na literatura encontramos muitas definições similares para roteamento, como a mostrada abaixo:

> O roteamento é responsável pela definição das rotas e dos materiais para a conexão de pinos de elementos que devem ter o mesmo potencial elétrico. Os elementos podem ser desde transistores isolados, pequenas células lógicas, macro células funcionais, grandes blocos já projetados, até componentes inteiros já empacotados. Os materiais disponíveis para roteamento são um conjunto de camadas metálicas, e furos nas camadas de isolamento para pôr em contato trechos de camadas metálicas verticalmente adjacentes. As conexões não podem se tocar quando não pertencem ao mesmo conjunto equipotencial de terminais, e, portanto, são obstáculos naturais entre si. Obstáculos externos são impostos também pela forma de implementação ou estilo de leiaute, tanto como limitação na área, como na presença de objetos dentro dela.(JOHANN, 2001).

Por ser um problema NP-Completo, muitas das estratégias para roteamento são heurísticas por natureza. Nas seções seguintes, são mostradas as principais e mais bem sucedidas técnicas utilizadas.

2.1 Técnicas Básicas

Em (SHERWANI, 1998) os algoritmos para roteamento global são classificados em duas categorias: abordagem sequencial e concorrente.

Na abordagem sequencial, como o nome sugere, as redes são roteadas uma a uma. Obviamente, redes roteadas antes tornam-se bloqueio para as posteriores, tornando esta abordagem fortemente dependente do ordenamento das redes. Em geral, as redes são ordenadas considerando sua criticalidade, semi perímetro e número de terminais. Esse sequenciamento, entretanto, não resolve satisfatoriamente o problema de ordenamento. Um aperfeiçoamento desta abordagem envolve a técnica de "*rip-up and reroute*"(DEES JR.; KARGER, 1982). Nesta técnica, as redes em conflito, ou seja, que estão ocupando um mesmo recurso, são desfeitas e re-roteadas. Outra técnica com o mesmo objetivo é deslocar as redes que interferem (impedem) outra rede, sem desfazer suas conexões. Outra abordagem é fazer o roteamento primeiramente das redes mais simples (com dois ou três terminais), já que estas têm menor flexibilidade (essa redes geralmente representam em torno de até 75% do total), sendo depois aplicados algoritmos de árvores de Steiner e *maze routing* nas redes restantes.

Nas sub-seções a seguir serão tratadas individualmente as técnicas mais difundidas e relevantes para a área. Nas seções seguintes vemos as aplicações dessas técnicas básicas em diversas implementações juntamente com outras técnicas e formas de implementação.

2.1.1 Roteamento de Labirinto (*Maze Routing*)

Os algoritmos de roteamento de labirinto são algoritmos de busca baseados em grade que têm a reputação de serem abordagens de força bruta para roteamento, no qual se conectam pares de pinos fonte e destino usando o menor caminho possível (de acordo com uma função custo arbitrária).

Algumas implementações não muito eficientes empregam pesquisa em largura (*Breadth First Search* - BFS) ou o algoritmo de Dijkstra. Uma abordagem melhor é o uso do algoritmo A* definido em (HART; NILSSON; RAPHAEL, 1968) e também de métodos de pesquisa bidirecional, como LCS* (*Lowerbound Cooperative Search*) apresentado em (JOHANN et al., 2000a,b).

Este tipo de algoritmo é a base dos roteadores detalhados e geralmente é o responsável pelo re-roteamento das redes em roteadores globais.

2.1.2 Roteamento Monotônico

O roteamento monotônico (PAN; CHU, 2007) é uma forma mais restrita de *maze routing*. Neste caso, a busca só é feita na direção do destino, ou seja, apenas os pontos da grade que ficam dentro do perímetro definido pelos dois pinos (origem e destino) são expandidos, reduzindo assim o espaço de busca e consequentemente reduzindo o tempo de execução.

2.1.3 Roteamento de Formas Padrão (*Pattern Routing*)

O roteamento de formas padronizadas (KASTNER; BOZORGZADEH; SARRAFZADEH, 2002) considera um número de caminhos muito menor que o *maze routing*, com o objetivo de aumentar a velocidade do algoritmo e torná-lo mais previsível. Neste algoritmo formas padrões (em formato de "L"e "Z") são consideradas para o roteamento entre os pinos, diminuindo consideravelmente o número de arestas da grade a serem examinadas. Tal algoritmo não garante a excelência do caminho escolhido e tipicamente precisa ser usado em conjunto com outro algoritmo (em geral *maze routing*) para gerar soluções com uma qualidade adequada.

2.1.4 Construção de Árvores de Steiner

Ao contrário dos métodos descritos nas sub-seções anteriores, que são inicialmente projetados para conexão de apenas dois pinos, a construção de árvores de Steiner[1] é projetada para redes multi-terminal. Como na maioria dos problemas de roteamento de circuitos VLSI os fios devem ter direção vertical ou horizontal, em geral, considera-se somente árvores de Steiner retilíneas (*rectilinear Steiner tree* - RST), mais comumente, busca-se usar a construção de menor comprimento de fio (*rectilinear Steiner minimum tree* - RSMT), que é sabidamente um problema NP-Completo.

Muitas técnicas foram propostas usando árvores de Steiner (BOESE et al., 1995; CONG; KOH, 1997; CONG; LEUNG; ZHOU, 1993; HOU; HU; SAPATNEKAR, 1999; HU; SAPATNEKAR, 2000a; LILLIS et al., 1996; VITTAL; MAREK-SADOWSKA, 1994;

[1] Árvores que incluem terminais adicionais, chamados nodos de Steiner, introduzidos para reduzir o comprimento total de fio.

ZHOU, 2004; BORAH; OWENS; IRWIN, 1994; YILDIZ; MADDEN, 2002) e um algoritmo exato, conhecido como GeoSteiner, também foi apresentado (WARME; WINTER; ZACHARIASEN, 2001). A maioria destes algoritmos focam na otimização de uma rede simples, desconsiderando o congestionamento. Tais algoritmos podem ser usados para roteamento serial de redes, começando pelas redes mais críticas. Outra opção é considerar o congestionamento como parte do custo das arestas. Entretanto, o maior uso destes algoritmos é para a geração de soluções iniciais de roteadores globais e para estimativas de roteamento na fase de posicionamento.

2.1.5 Programação Linear Inteira 0-1

O roteamento global pode ser formulado como um tipo especial de problema de otimização, conhecido como programação linear inteira 0-1 (*zero-one Integer Linear Programming* - 0-1 ILP).

Na prática, como o problema 0-1 ILP é NP-Completo, usa-se técnicas diferentes para resolução deste problema, mas mesmo assim, dificilmente usa-se essa técnica para resolver o problema inteiro de roteamento global, sendo mais frequentemente usada dentro de outra estratégia de roteamento que divide o problema hierarquicamente, até um nível onde computar a solução ótima 0-1 ILP seja possível. Essa técnica é usada progressiva e incrementalmente no roteador global BoxRouter (CHO; PAN, 2007; CHO et al., 2007) que é descrito na seção 3.1.2 e também no GRIP (WU; DAVOODI; LINDEROTH, 2009, 2010, 2011) e no Sidewinder (HU; ROY; MARKOV, 2008).

2.1.6 Modelo de Fluxo de Rede

É intuitivo associar o problema de roteamento global com um modelo de fluxo de rede qualquer (redes de transportes e de dados, por exemplo). Num modelo básico de fluxo, existem dois vértices especiais, chamados origem e destino. Uma certa quantidade do fluxo, chamada demanda, deve ser entregue da origem para o destino. Cada aresta tem uma capacidade de fluxo, que representa o fluxo máximo que pode passar por esta aresta. Existem duas versões para o problema, sendo que uma delas requer o transporte do máximo possível de fluxo desde que respeitadas as capacidades das arestas, chamada de problema de fluxo máximo. A outra versão assinala um custo por unidade de fluxo para cada aresta, e o objetivo do problema é minimizar o custo total de transporte pela rede para um determinado fluxo a partir da origem, sendo chamado de problema de fluxo de mínimo custo.

Esse modelo é usado por poder ser resolvido em tempo polinomial, para capacidades de arestas inteiras. Apesar de não poder ser modelada para todo o problema de roteamento global, esta técnica (*commodity* única) pode ser empregada para resolver subproblemas em roteamento global (CHO; SARRAFZADEH, 1998; HU; SAPATNEKAR, 2000b; MEIXNER; LAUTHER, 1990) e pode levar a soluções de alta qualidade.

Existe um tipo especial de modelo de fluxo de rede que pode ser diretamente aplicado a roteamento global, chamado de problema de fluxo *multicommodity*, onde várias *commodities* devem ser entregues numa rede em comum, e cada *commodity* tem suas próprias origens e destinos. Assim, no roteamento global, cada rede pode ser tratada como uma *commodity*.

Uma vantagem desta abordagem é que este problema pode ser formulado como um problema de programação linear. Como os resolvedores de programas lineares são lentos para a ordem de grandeza dos problemas de roteamento global, a pesquisa sobre esse tema é mais focada em algoritmos heurísticos e de aproximação combinatória, que serão

discutidos mais adiante na seção 2.6.

2.2 Técnicas de Roteamento Sequencial

Uma forma de fazer o roteamento de múltiplas redes é definir uma ordem para estas redes e então roteá-las sequencialmente, considerando as redes já roteadas como obstáculos. Como citado anteriormente, essa abordagem gera resultados muito dependentes do ordenamento das redes, sendo um bom ordenamento difícil de ser encontrado, e também difícil, sob qualquer ordenamento, rotear as redes que ficam para o final, já que estas estão sujeitas a mais bloqueios (redes já roteadas).

Apesar desses fatores, existem diversos trabalhos que apresentam bons resultados usando roteamento sequencial, sendo que a maioria usa laços iterativos que informam o congestionamento das redes roteadas mais tarde para as redes previamente roteadas.

2.2.1 Roteamento Baseado em Forças (*Force-Directed*)

Outra forma de modelamento do problema de roteamento global para redes de dois pinos é emular o movimento de partículas em um campo de força (HASAN; LIU, 1987). As redes são roteadas sequencialmente, sendo que para cada rede uma partícula enviada do pino de origem até o pino de destino, sendo estimulada pelas forças do pino de origem, de destino, pinos não roteados de outras redes e fios de redes já roteadas. A trajetória da partícula é então o caminho de roteamento.

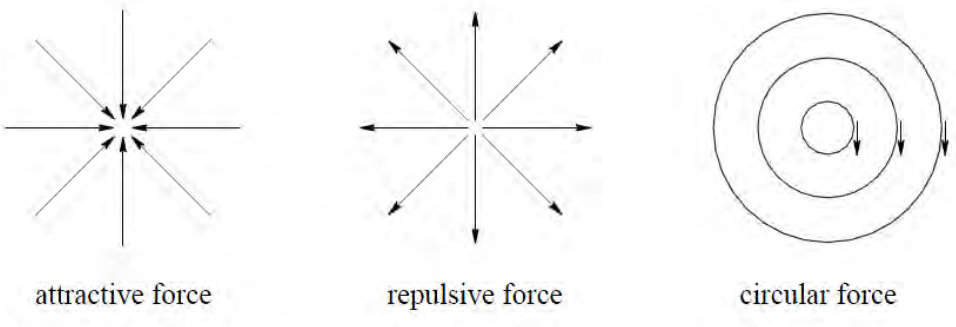

Figura 2.1: Modelamento das forças. (HU; SAPATNEKAR, 2001)

Existem três tipos de forças, conforme a Figura 2.1. Na Figura 2.2 vemos como as forças são exercidas, sendo que em (a) temos a força exercida por um pino não roteado, que está em linha e coluna diferente da partícula, exercendo portanto uma força repulsiva, definida como de magnitude $1/pr^2$, onde p é o número total de pinos não roteados e r a distância entre a partícula e o pino não roteado. O pino de origem também gera uma força repulsiva, só que de magnitude $1/r^2$. Já o pino de destino é o único a exercer força de atração, de magnitude $1/r^{1.5}$, que em geral é maior que qualquer outra força, garantindo assim que a partícula chegue ao destino. Pinos não roteados ou segmentos de fio que estão na mesma linha ou coluna que a partícula exercem força circular de magnitude $1/r^2$, sempre na mesma direção das forças de fonte e destino, como mostrado na Figura 2.2(b).

Este método de forças pode ser paralelizado fazendo com que cada partícula para todas as redes movam-se simultaneamente.

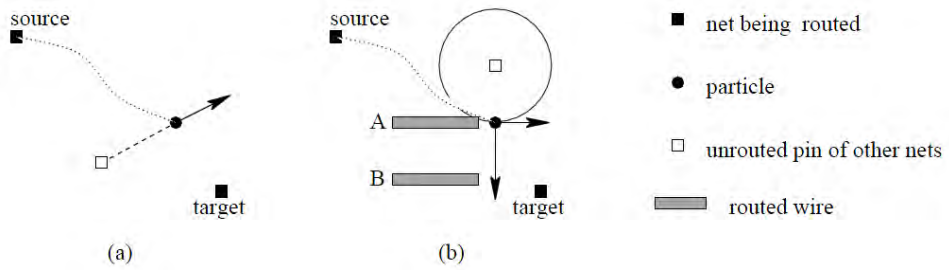

Figura 2.2: Exemplo de forças exercidas sobre uma partícula: (a) força repulsiva de um pino não roteado numa coluna e linha diferentes da partícula. (b) uma força circular de um pino não roteado na mesma linha ou coluna da partícula. (HU; SAPATNEKAR, 2001)

2.2.2 Roteamento Sequencial por Construção de Árvores de Steiner min-max

Uma árvore de Steiner min-max (SMMT) é uma árvore de Steiner na qual o peso máximo das arestas é minimizado para todas as árvores de Steiner. Em (CHIANG; SARRAFZADEH; WONG, 1990) o problema de roteamento global é solucionado utilizando a construção de SMMTs para cada rede sequencialmente, definindo os pesos das arestas para estimar o congestionamento, sendo roteadas as redes de menor perímetro primeiro.

A SMMT ótima é obtida em duas etapas de tempo polinomial, sendo que na primeira etapa é construída a MST para todas as células do grafo de roteamento e na segunda etapa todas as células não-terminais (que não possuem pinos da rede em questão) de grau um são removidas da MST, resultando na SMMT ótima. Os pesos das arestas são atualizados dinamicamente.

Este procedimento claramente busca diminuir o congestionamento (peso das arestas), entretanto ele não considera o comprimento dos fios ao fazer isso. Tal problema, encontrar uma SMMT com mínimo comprimento de fio (MSMMT), é NP-Completo. Uma heurística para resolução deste problema também é apresentada em (CHIANG; SARRAFZADEH; WONG, 1990).

2.2.3 Árvores de Steiner de Peso Mínimo

Outro método sequencial que busca minimizar simultaneamente congestionamento e comprimento de fio é descrito em (CHIANG; WONG; SARRAFZADEH, 1994), que executa o roteamento das redes usando um algoritmo aproximado de geração de árvores de Steiner retilíneas de peso mínimo (MWRST). Neste método a área do circuito é dividida em pequenas regiões, cada qual com um peso baseado na complexidade e congestionamento da região, como mostrado no exemplo da Figura 2.3 (a). Os retângulos sombreados representam regiões de bloqueio, ou seja, peso infinito. É então construído o grafo de roteamento para uma rede através da união da grade de Hanan para esta rede e a grade formada pela extensão das bordas de cada região até que esta encontre um bloqueio ou a borda do circuito - Figura 2.3(b). O peso da aresta é então calculado como a multiplicação do comprimento da aresta pelo peso da área na qual ela se encontra. O objetivo do algoritmo é minimizar este valor.

Como no método anterior, as redes são ordenadas em ordem crescente de perímetro, sendo os pesos das regiões atualizados a cada rede roteada.

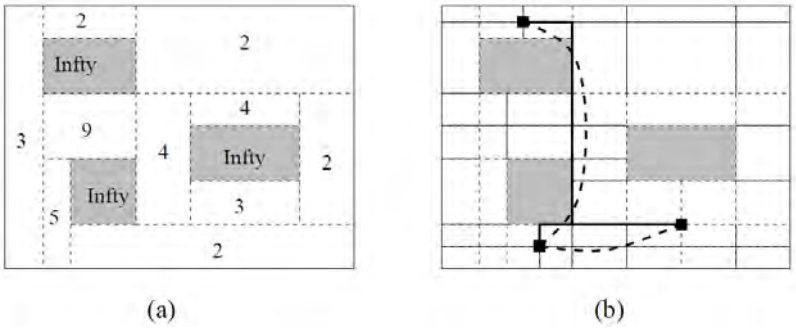

Figura 2.3: (a) regiões com diferentes pesos em um leiaute. (b) grafo de roteamento, MST (linha pontilhada) e WRST (linha sólida) para este leiaute. (HU; SAPATNEKAR, 2001)

2.3 Meta-heurísticas

As heurísticas baseadas em movimento são comumente usadas para encontrar soluções ótimas (ou o mais próximo possível) para problemas computacionalmente difíceis. Dentre estes métodos o mais conhecido está o *simulated annealing* (ou recozimento simulado) (KIRKPATRICK; GELATT JR.; VECCHI, 1983). A principal motivação deste método é que heurísticas gulosas para tratamento de problemas de otimização combinatória podem facilmente cair e ficar presos em mínimos locais, pois apenas movimentações que provocam ganho no custo são aceitas. Este método é amplamente conhecido em CAD e não será detalhado aqui.

A técnica de *simulated annealing* é aplicada em roteamento global em (VECCHI; KIRKPATRICK, 1983), onde o número de pinos por rede é de apenas dois e as dobras de cada rede não são maiores que duas.

No pacote TimberWolf (SECHEN; SANGIOVANNI-VINCENTELLI, 1986), a técnica de *simulated annealing* é aplicada para posicionamento e roteamento global. Neste pacote um conjunto de árvores candidatas é criado para cada rede, sendo uma escolhido aleatoriamente como a inicial, e cada movimento representa a troca de uma árvore candidata por outra para uma determinada rede, sendo o *overflow* o custo a ser minimizado. A rede a sofrer a troca é escolhida de uma célula cuja fronteira tenha *overflow*.

Outras técnicas baseadas em movimento também são aplicadas ao roteamento global, como evolução simulada (CHEN; LIN; HSU, 1989), algoritmos genéticos (ESBENSEN, 1994) e pesquisa tabu (YOUSSEF; SAIT, 1999).

2.4 *Rip-up and Reroute*

Outra abordagem comum usada para contornar o problema de ordenamento de redes é o método de *rip-up and reroute* (algo como desfaz e re-roteia). Neste método todas as redes são roteadas sem considerar congestionamento, usualmente construindo-se árvores de Steiner mínimas para cada rede. Após essa etapa, identifica-se as áreas onde há congestionamento e as redes que têm caminhos por essas áreas são desfeitas e roteadas novamente para regiões menos congestionadas, usando normalmente algum algoritmo *maze routing*.

Apesar de sua simplicidade, o método de *rip-up and reroute* é altamente efetivo e tem sido o método mais usado na indústria (HU; SAPATNEKAR, 2001). Outra qualidade deste método é que ele pode ser aplicado em conjunto com outros métodos, como um pós-processamento para melhorar a qualidade do roteamento. Os parâmetros que podem ser ajustados nesse método são, por exemplo, as diferentes estratégias para escolha da rede a

ser roteada novamente ou a ordem na qual as regiões com *overflow* serão processadas.

Em (TING; TIEN, 1983) o método de *rip-up and reroute* seleciona um conjunto de fronteiras de células (arestas no grafo de roteamento) congestionadas e então escolhe o subconjunto de redes que passam por essa fronteira para serem roteadas novamente.

2.5 Roteamento Baseado em Negociação (*Negotiation-based Routing*)

O roteamento baseado em negociação de congestionamento foi primeiramente proposto em (MCMURCHIE; EBELING, 1995) chamado *PathFinder*, com aplicação para FPGAs, mas que tem sido largamente usado em projetos mais genéricos. Neste esquema de negociação, que utiliza a técnica anterior de *rip-up and reroute*, os fios (sinais) podem inicialmente compartilhar os recursos de roteamento, mas depois devem negociar com os outros fios para determinar qual sinal precisa mais do recurso.

Neste trabalho, o roteador global ajusta um fator de penalização de congestionamento para cada recurso baseado na demanda de sinais daquele recurso. Na primeira iteração não há custo de compartilhamento, permitindo que os sinais passem pelos mesmos recursos. A partir da segunda iteração o fator de penalização é gradualmente incrementado conforme a sua demanda, fazendo com que os sinais negociem os recursos.

Assim os sinais que não tem alternativas melhores permanecem no recurso que está aumentando de custo enquanto que os sinais que possuem outras alternativas de roteamento espalham-se para recursos menos demandados. Desta forma, a cada iteração todas as redes que estão compartilhando recursos são desfeitas e re-roteadas com base nos novos custos, até que mais nenhum recurso seja compartilhado.

Na Figura 2.4 há um exemplo de congestionamento a ser tratado, onde existem três sinais, com suas origens e destinos e três caminhos (ou recursos de roteamento) e seus caminhos parciais com seus respectivos custos. Neste caso, o melhor caminho para S_1 seria utilizando o recurso *B* e para os sinais S_2 e S_3 seria utilizando o recurso *C*. Para resolver este impasse, o algoritmo vai iterativamente incrementando o custo dos recursos compartilhados de acordo com a equação abaixo, até que os sinais escolham os caminhos possíveis (S_1 escolha *A* e S_2 escolha *B*).

$$c_n = (b_n + h_n) * p_n$$

onde b_n é o custo base (quantidade de recursos já utilizada), h_n o histórico de congestionamento do nodo n e p_n o número de sinais compartilhando n no momento.

No mesmo trabalho é apresentado uma forma de englobar também o atraso das interconexões nesta abordagem. A função custo recebe então também o custo de atraso, como definido na equação apresentada abaixo que usa a definição de c_n anterior, mudando o custo de um caminho de s_i até t_ij como sendo

$$C_n = A_{ij}\, d_n + (1 - A_{ij})\, c_n$$

onde

$$A_{ij} = D_{ij} / D_{max}$$

sendo D_{ij} o maior caminho contendo o arco (s_i,t_ij) e D_{max} o atraso de caminho crítico. Assim $0 < A_{ij} \leq 1$.

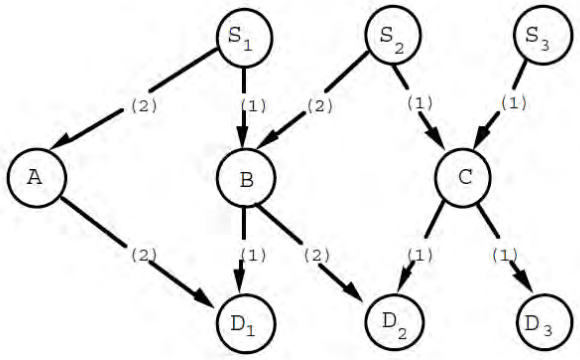

Figura 2.4: Exemplo de congestionamento, com custos entre parênteses. (MCMURCHIE; EBELING, 1995)

Neste caso, quando um caminho pertencente ao caminho crítico é tratado, $A_{ij} = 1$, ou seja, o custo de congestionamento é desconsiderado, dando-se prioridade para o caminho crítico ocupar o recurso.

A técnica apresentada neste trabalho tem sido amplamente utilizada, sendo a base de muitos roteadores globais atuais, como nos trabalhos de ZHANG; XU; CHU (2008); GAO; WU; WANG (2008); DAI; LIU; LI (2009); HSU; CHEN; CHANG (2008); HU; ROY; MARKOV (2010); OZDAL; WONG (2007); LIU et al. (2010); CHANG; LEE; WANG (2008); CHEN; HSU; CHANG (2009); CHO et al. (2007).

2.6 Abordagem Baseada em Fluxo Múltiplo de Mercadorias (*Multi-commodity Flow*)

Apesar das abordagens tratadas anteriormente serem efetivas na prática, elas não podem garantir quando existe ou não uma solução factível, já que não existe como deixar claro se o fato de não ter encontrado uma solução é por não haver uma ou pelos "atalhos"das heurísticas. Outro fato é que as heurísticas não sabem quando a solução é ótima ou quão distante da ótima ela se encontra.

Uma forma de cobrir esses espaços é formular o problema de roteamento global como um problema de fluxo múltiplo de mercadorias, onde um grafo *G* = (*V,E*), com *V* = $\{v_1, v_2, .., v_n\}$ sendo o conjunto de *n* vértices e *E* = $\{e_1, e_2, .., e_m\}$ o conjunto de *m* arestas representa uma rede pela qual deverá ser transportada uma quantidade *k* de mercadorias de alguns vértices para outros, exigindo uma certa demanda d_i.

Em roteamento global trata-se cada rede como uma mercadoria, então um conjunto de mercadorias *N* = $\{N_1, N_2, .., N_k\}$ precisa ser transportada pela rede (recursos de roteamento do CI), com uma demanda d_i igual a um. Cada aresta tem uma capacidade $u(e)$ e um custo $c(e)$. Através deste modelamento, utilizando-se 0-1 ILP e uma abordagem similar à descrita na seção 2.1.6, pode-se usar diferentes objetivos para o problema, de acordo com as características necessárias. Mais detalhes sobre os diferentes algoritmos propostos na literatura são encontrados em HU; SAPATNEKAR (2001), sendo o trabalho mais recente de ALBRECHT (2001) (que também utiliza uma formulação baseada em ILP), que mostrou-se mais lento e com soluções de menor qualidade que a abordagem usando ILP (CHO; PAN, 2007).

2.7 Métodos Hierárquicos

A ideia central dos métodos hierárquicos, cujo primeiro trabalho foi proposto por BURSTEIN; PELAVIN (1983), é usar a conhecida técnica de "dividir para conquistar", onde divide-se as regiões de roteamento sucessivamente em regiões menores, transformando o complicado problema de roteamento numa série de problemas mais simples, tornando o roteamento muito mais rápido. Obviamente, a simplicidade deste método apresenta suas limitações quanto à qualidade do roteamento, já que decisões ruins são levadas para níveis subsequentes, degradando a qualidade da solução. Por isso, tal método geralmente é utilizado com outras técnicas, como a já descrita ténica de *rip-up and reroute*.

Vários trabalhos da literatura apresentam diferentes técnicas para implementar este método, sendo as principais abordagens descritas nas subseções a seguir.

Refinamento Sucessivo Top-down: proposto inicialmente em BURSTEIN; PELAVIN (1983), este método baseia-se na redução sucessiva das áreas de roteamento, como mostrado na Figura 2.5, até que as células fiquem reduzidas ao tamanho das GRCs. No mesmo trabalho são propostas heurísticas para a solução do problema visando minimizar o congestionamento e o comprimento de fio.

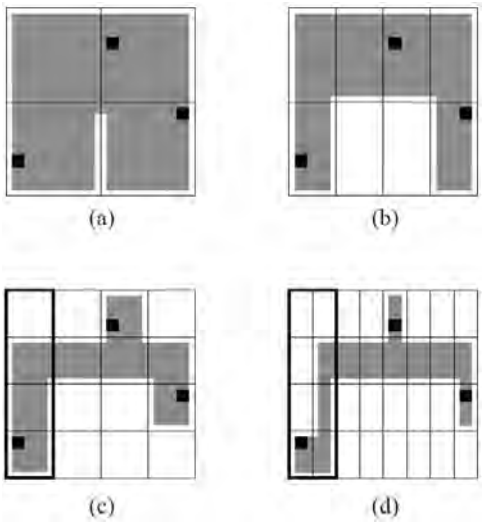

Figura 2.5: Exemplo de refinamento sucessivo *top-down*. (HU; SAPATNEKAR, 2001)

União Bottom-up: este método é proposto em LI; MAREK-SADOWSKA (1984) como uma evolução para fugir das limitações do método anterior. Neste método a divisão das células começa pelo nível hierárquico mais baixo, onde as GRCs são agrupadas em arranjos 2x2, sendo as redes dentro deste arranjo roteadas, não podendo cruzar as fronteiras do arranjo. Finalizada a etapa sobe-se o nível, transformando cada arranjo em uma célula, criando novos arranjos 2x2, repetindo até que toda a área de roteamento seja um único arranjo 2x2, como mostrado na Figura 2.6. Este método é seguido pelo processo de *rip-up and reroute*. Métodos *bottom-up* baseados em programação linear também são encontrados na literatura (HU; SHING, 1985).

Método Hierárquico Híbrido: este método associa os dois métodos anteriores para evitar que decisões em um nível sejam prejudiciais para o próximo nível (HAYASHI; TSUKIYAMA, 1995). O algoritmo consiste de dois laços de repetição, um hierárquico *top-down* dentro de outro hierárquico *bottom-up*, sendo usadas dentro do laço mais interno as técnicas de *maze routing* e *rip-up and reroute*.

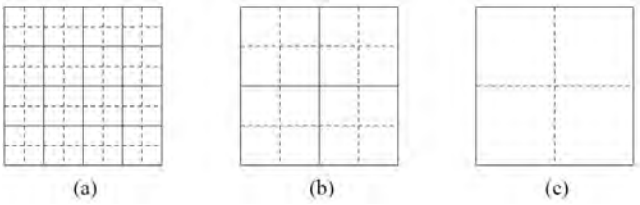

Figura 2.6: Exemplo de união *bottom-up*. (HU; SAPATNEKAR, 2001)

Roteamento Hierárquico para custom design: como os métodos anteriores baseiam-se em um grafo de grade uniforme, exitem também trabalhos para grades não uniformes, baseando-se na planta baixa do circuito (LUK et al., 1987). O algoritmo divide toda a região em quatro células, as quais respeitam as fronteiras da planta baixa e, portanto, englobam todos os blocos definidos, roteando as células como se os pinos estivessem no centro das mesmas. A cada iteração as quatro regiões são divididas em quatro, sendo isso repetido até que cada região seja um bloco da planta baixa.

Bi-secção Hierárquica e Assinalamento Linear: este método difere-se dos anteriores por não fazer a divisão em quatro secções, mas sim em duas (MAREK-SADOWSKA, 1986), podendo ser aplicado para grade regular e não regular. Também é um método *top-down*.

Roteamento Hierárquico de Quatro Dobras: este método é definido por CHO; SARRAFZADEH (1998), onde também divide-se toda a área em regiões 2x2 num processo *top-down*. Ele é dividido em dois estágios, sendo que no primeiro é definido em quais regiões de corte as redes devem atravessar (utiliza-se programação inteira para resolver este problema) e no segundo determina-se qual canal a rede irá atravessar (formulado como um problema de fluxo de rede de custo mínimo).

2.8 Roteamento Global Dirigido a Desempenho

Como o atraso devido às interconexões tem sido determinante no desempenho dos CIs, considerar apenas o congestionamento durante a fase de roteamento global não é mais adequado. Devido a este fato, vários trabalhos apresentam e incorporam as questões de atraso dentro do roteamento global (REIMANN; SANTOS; REIS, 2010; CONG; MADDEN, 1997; HONG et al., 1997; HU; SAPATNEKAR, 2000b; HUANG et al., 1993; WANG; KUH, 1996; YOUSSEF; SAIT, 1999; XU; HONG; JING, 2005; XU et al., 2003, 2004, 2002).

No trabalho de HONG et al. (1997) vemos uma abordagem baseada em fluxo múltiplo de mercadorias (*multicommodity flow*, onde foi desenvolvido o roteador global chamado TIGER, que adota um fluxo similar ao do trabalho de (CARDEN R.C.; LI; CHENG, 1996), incorporando a questão de atraso ao congestionamento.

Em (CONG; MADDEN, 1997) encontramos um método de roteamento global dirigido a desempenho que inicialmente constrói MSTs para cada rede e caso esta MST não satisfaça a limitação de tempo (*timing constraint*), uma topologia dirigida a desempenho é construída para esta rede e técnicas de dimensionamento de fio são aplicadas para garantir a limitação de tempo. Após isso, as redes críticas têm sua topologia fixada e um grafo de conexão simplificado é construído para as outras redes. Para minimizar o congestionamento é executado um último passo de deleção iterativa.

Já em (HU; SAPATNEKAR, 2000b) foi desenvolvido uma abordagem onde após um roteamento inicial dirigido a desempenho, para que as redes atinjam o objetivo de atraso,

o congestionamento é reduzido através de processos de bi-secção hierárquica e assinalamento, similares aos de MAREK-SADOWSKA (1986) citados anteriormente.

2.9 Roteamento Global Dirigido a *Crosstalk*

O problema de *crosstalk* entre linhas de interconexão tem se tornado um dos grandes problemas em tecnologias atuais. Tal problema é mais comumente tratado na fase de roteamento detalhado, mas como este está limitado à pequenas regiões, nem sempre é possível resolver o problema neste estágio. Em (ZHOU; WONG, 1999) explora-se a ideia de evitar o *crosstalk* já no roteamento global, explorando a maior flexibilidade de movimentação das redes deste estágio para aliviar este problema.

Obviamente, na fase de roteamento global ainda não existe a informação de adjacência das linhas de metal. Para resolver isso, o trabalho citado faz um assinalamento de camadas e linhas *on the fly* para obter estimativas de capacitâncias de acoplamento. Neste algoritmo é feito então o cálculo de *crosstalk* para cada rede adicionado no custo total, sendo assim, cada rede que possui uma violação é desfeita e re-roteada para satisfazer a limitação de *crosstalk* usando um método baseado na relaxação Lagrangiana.

Outros trabalhos mais recentes também podem ser encontrados, apresentando uma gama diferente de técnicas para diminuição dos problemas de acoplamento (XU et al., 2003; XIONG; HE, 2005; MA; HE, 2002; ZHANG et al., 2004; JING et al., 2005; XU et al., 2004).

3 CONCURSOS SOBRE ROTEAMENTO GLOBAL E TRABALHOS RECENTES

Um dos fatores que alavancou o desenvolvimento recente no que diz respeito ao roteamento global foram os dois concursos realizados pelo ISPD nos anos de 2007 e 2008. Tais concursos trouxeram novamente o foco para o problema de roteamento global, entretanto, nos dois anos de realização do concurso só foi abordado o congestionamento como métrica principal de avaliação dos roteadores globais. Como dito anteriormente, esta não é mais o único objetivo de um roteador global, mas sim alcançar os objetivos de desempenho que cada vez mais dependem das interconexões.

Apesar disso, os participantes dos concursos apresentaram novas e interessantes técnicas de roteamento que serão detalhadas nas seções a seguir, dando ênfase para os roteadores que obtiveram os melhores desempenhos.

3.1 ISPD *Contest* 2007

A grande novidade trazida por este concurso foi a introdução de circuitos de teste (*benchmarks*) tridimensionais (3D), ou seja, que não consideram o problema de roteamento como um grafo planar (2D) mas sim como um problema 3D, onde são consideradas as diferentes camadas de metal com diferentes capacidades de roteamento, devido às diferentes larguras de fio nas camadas.

Apesar disso, a maioria dos roteadores não usa a construção de grafo 3D para a resolução do problema, fazendo um mapeamento de capacidades para 2D, o que não afeta a qualidade das soluções, já que não há restrições ao uso de vias, como será visto nas próximas subseções, onde serão discutidos os principais aspectos dos roteadores vencedores do concurso.

A forma de cálculo do comprimento de fio é como mostrado na equação abaixo:

$$WL = \sum_{n \in N}^{n} wl(n) + peso\_via \times via(n)$$

onde o peso das vias é igual a 3 e $wl(n)$ e $via(n)$ representam o comprimento de fio e o número de vias de uma rede n, respectivamente.

3.1.1 FGR - *Fairly Good Router*

O trabalho apresentado por ROY; MARKOV (2007, 2008) tem como principal objetivo adequar o custo de roteamento para balancear o comprimento das interconexões e congestionamento em projetos com vários milhões de portas lógicas de tal forma que

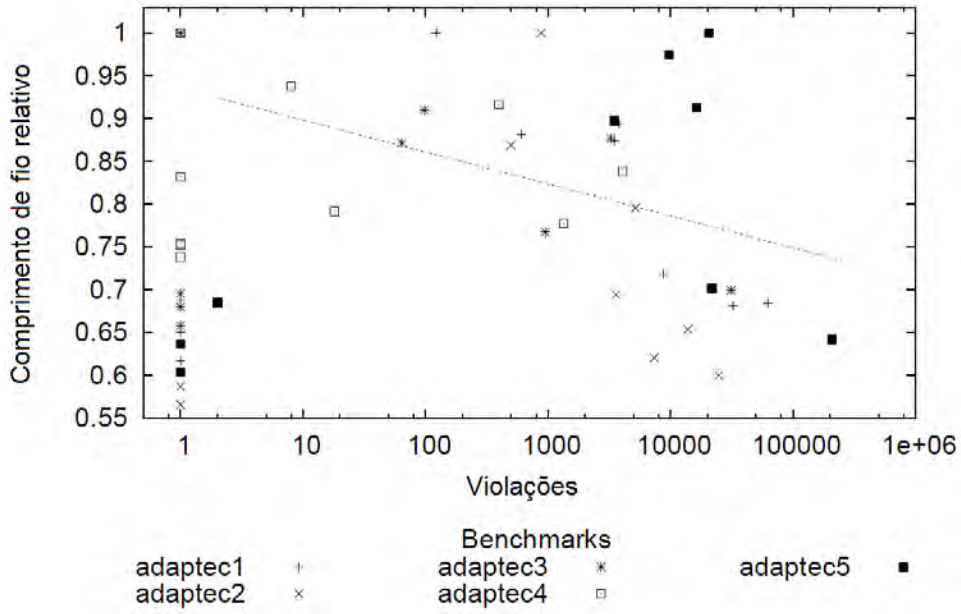

Figura 3.1: Comprimento de fio relativo versus violações para todos roteadores nos *benchmarks* 2D do ISPD 2007 *Contest*. Adaptada de (ROY; MARKOV, 2008)

permita também o compromisso com outros objetivos e limitações impostas nos projetos de escala nanométrica. A Figura 3.1 mostra que os roteadores que apresentam menor comprimento de fio são os que apresentam mais violações[1], enfatizando o objetivo do FGR.

Os principais desafios destes projetos são: a grande quantidade de fios que precisam ser armazenados numa estrutura de dados eficiente, sofisticadas regras de projeto que precisam ser abstraídas no roteamento global, vias inconstantes (com valores de resistência que chegam a variar 30x) que requerem redundância, limitações com relação à integridade de sinal devido às capacitâncias laterais e limitações de densidade de metal que provocam diferentes características de fios devido ao processo de polimento químico-mecânico (CMP).

Neste trabalho é apresentada uma técnica de roteamento baseada nos Multiplicadores de Lagrange (DLM) que proveem uma forma natural de lidar com os pesos das redes e otimização de desempenho em roteamento global, usando-se tal técnica associada às de *rip-up and reroute*, decomposição de redes por RMST ou RSMT e A*-*search*. Tal método permite ao algoritmo tratar problemas de roteamento de ASICs com até um milhão de redes (com espaço de endereçamento de 32 bits).

O método dos multiplicadores de Lagrange é usado para encontrar o mínimo de uma função de comprimento de fio suscetível a uma ou mais restrições, no caso, a capacidade das GRCs (*overflow*). Assim, para roteamento global, tem-se:

$$\min_{\mathbf{x}\in X} W(\mathbf{x})$$
$$\text{desde que } C_e(\mathbf{x}) = 0, \quad 1 \leq e \leq n$$

Mapeando-se W(x) como sendo os recursos de roteamento de cada aresta o problema é reduzido para um problema de otimização da função de Lagrange F abaixo:

[1]Como está em escala logarítmica, zero violações são representadas como uma.

$$F(\mathbf{x}, \lambda) = \sum_{e=1}^{n} \left(B_e(\mathbf{x}) + \lambda_e C_e(\mathbf{x}) \right)$$

onde $B_e(x)$ é o número de redes passando pela aresta e, $C_e(x)$ o *overflow* da aresta e e $\lambda = (\lambda_1, ..., \lambda_n)$ os multiplicadores de Lagrange. A incógnita x e os multiplicadores de Lagrange são consideradas variáveis a serem otimizadas. Nesse trabalho os multiplicadores são atualizados conforme a equação abaixo:

$$\lambda^{k+1} = \lambda^k + \alpha C(\mathbf{x}^k)$$

onde $\alpha > 0$ é o parâmetro de linha de busca. Essa atualização é similar à atualização de h_n usada no método de roteamento baseado em negociação da seção 2.5. Assim, interpretando-se $F(x, \lambda)$ em termos do método baseado em negociação tem-se:

$$c_e = b_e + h_e \cdot p_e$$

Para o termo p_e, o FGR usa a seguinte definição:

$$p_e = \begin{cases} \exp\left(k(\omega_e - 1)\right) & \text{se } \omega_e > 1 \\ \omega_e & \text{, caso contrário} \end{cases}$$

onde w_e é o *overflow* relativo e k é uma constante que com valores maiores resulta em soluções mais rápidas mas com mais desvios e, portanto, comprimento de fio maior. O FGR usa $k = ln5$, representada na Figura 3.2.

Em vez de gerar uma solução inicial de menor caminho para todas as redes, o FGR usa $b_e + p_e$ como peso das arestas para criar a solução inicial. As redes são re-roteadas durante a fase de cálculo do DLM com uma versão modificada do A*-*search*. Essa modificação é na função custo usada durante a execução do algoritmo, onde é adicionada uma checagem que verifica se a aresta em consideração já pertence a outra sub-rede da mesma rede (todas as redes são decompostas em redes de dois pinos) então o custo da aresta é temporariamente zerado, permitindo que o *maze router* mude a topologia da rede.

Outro fator importante é a construção da topologia da rede. Tradicionalmente a decomposição de redes é feita usando árvores de expansão mínimas (*Minimal Spanning Tree* - MST), mas com o aumento de eficiência dos algoritmos de geração de RSMTs estes passaram a ter grande uso também (CHO; PAN, 2007; PAN; CHU, 2006, 2007; ZHANG; XU; CHU, 2008). O FGR pode usar os dois métodos, e os resultados de cada um deles são mostrados na Tabela 3.1.

Nesta comparação é usado MST e uma combinação de RSMTs geradas pelo FLUTE (CHU; WONG, 2008) e pelo FastSteiner (KAHNG; MANDOIU; ZELIKOVSKY, 2003) que retorna sempre a melhor árvore de Steiner. Como a mudança na topologia da rede pode ocorrer durante o roteamento de cada segmento, o FGR produz árvores de Steiner não triviais mesmo quando usa-se a decomposição por MST.

O custo e a variabilidade das vias têm se tornado muito altos nas tecnologias atuais, tornando a minimização das vias (para posterior replicação para melhorar o *yield*) um passo importante durante a síntese física. Uma forma de adequar o uso das vias é especificar corretamente o seu custo. Para fazer isso o FGR trata as vias como arestas no grafo de roteamento, o que possibilita que estas tenham custo diferente das demais arestas, pois

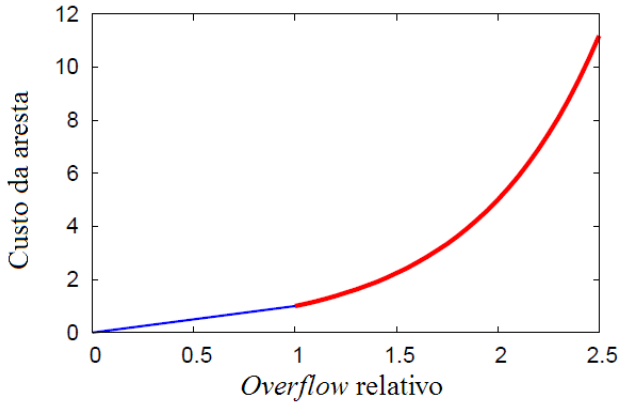

Figura 3.2: Função custo vs. *overflow* de uma aresta. Adaptada de (ROY; MARKOV, 2008)

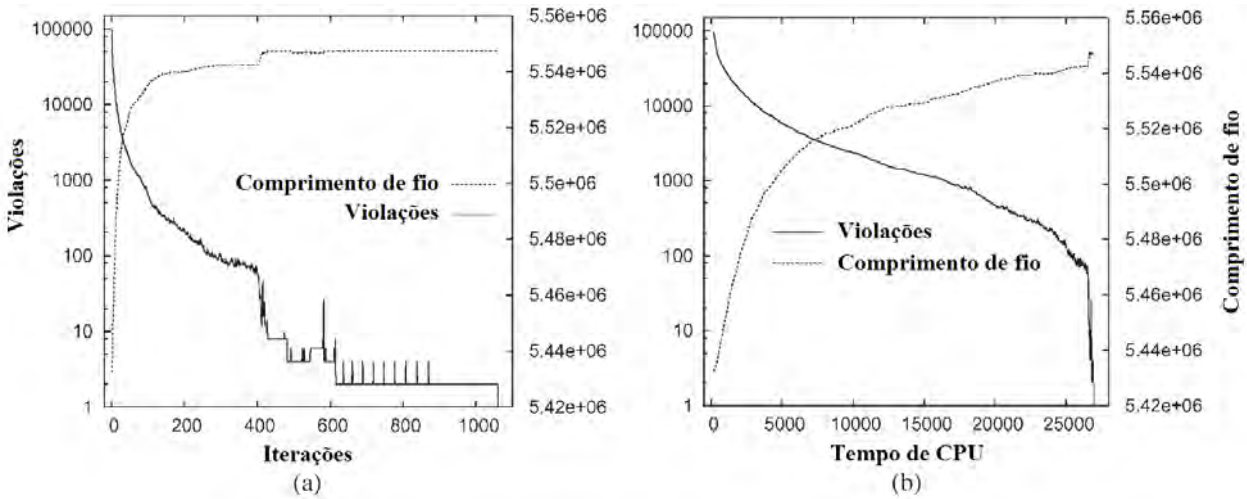

Figura 3.3: Violações e comprimento de fio em função do (a) número de iterações e do (b) tempo de CPU para o *benchmark* adaptec1 2D. Adaptada de (ROY; MARKOV, 2008).

a resistividade das vias é muito maior que a das linhas de metal. A Tabela 3.2 mostra a quantidade de vias e o custo delas relativo ao custo total para os *benchmarks* do ISPD 2007, onde uma via equivale a três segmentos da grade de roteamento, sendo então o custo base das arestas das vias $3b_e$, fazendo com que o A* naturalmente otimize o uso das vias na busca pelo menor caminho.

Apesar do uso de DLM se comportar bem para problemas grandes, quando restam poucas violações no circuito, constatou-se que o FGR gastava muitas iterações de DLM (75% do tempo total, para apenas 0,01% de segmentos com *overflow*, para o circuito adaptec2). Esse comportamento foi denominado de "último suspiro"(*last-gasp*) e pode ser visto na Figura 3.3.

Para solucionar este problema foi proposto que quando a porcentagem de arestas de roteamento com *overflow* for pequena, restringe-se o *maze router* a usar apenas arestas que possuem recursos disponíveis e atribuindo os pesos das arestas apenas pelo custo base b_e. Assim, se houver uma forma de rotear a rede sem causar *overflow*, essa será a escolha, para evitar mais iterações de *rip-up*, caso contrário, usa-se DLM normalmente. Na maioria dos casos esta última fase de DLM reduz o número de iterações sem impacto

Tabela 3.1: Comparação da decomposição das redes por MST e árvore de Steiner para os *benchmarks* do ISPD 2007.

Benchmark	Decomposição por MST				Decomposição por árvore de Steiner			
	WL (e5)	Vias (e5)	Custo total	Tempo (min)	WL (e5)	Vias (e5)	Custo total	Tempo (min)
adaptec1 2D	35,88	6,19	54,44	451	35,78	6,24	54,49	403
adaptec1 3D	36,37	17,36	88,45	430	36,26	18,04	90,37	395
adaptec2 2D	33,21	6,36	52,30	56	33,10	6,43	52,38	170
adaptec2 3D	33,74	18,72	89,89	64	33,62	19,37	91,72	168
adaptec3 2D	96,09	11,60	130,89	179	95,55	11,67	130,57	222
adaptec3 3D	97,02	34,21	199,66	243	96,42	35,46	202,90	281
adaptec4 2D	90,02	11,66	125,00	19	89,37	11,72	124,53	18
adaptec4 3D	91,28	30,56	182,96	55	90,59	31,59	185,35	58
adaptec5 2D	102,79	16,45	152,13	713	102,56	16,63	152,45	771
adaptec5 3D	103,89	52,03	259,98	740	103,62	53,78	264,97	796
newblue1 2D	24,15	7,76	47,42	1441	24,00	7,74	47,22	1441
newblue1 3D	24,15	23,37	94,26	1442	24,00	24,00	96,01	1442
newblue2 2D	46,81	9,90	76,51	4	46,41	9,95	76,27	4
newblue2 3D	47,91	28,08	132,16	10	47,51	29,08	134,75	10
newblue3 2D	75,63	11,20	109,23	1555	75,24	11,15	108,71	1460
newblue3 3D	75,63	32,69	173,71	1501	75,24	33,04	174,35	1462
Relação					-0,52%	+1,81%	+0,74%	+22,0%

Tabela 3.2: Resultados de comprimento de fio para todos os circuitos.

Benchmark	WL (e5)	Seg/FLUTE	Vias (e5)	Custo total (e5)	Custo das vias
adaptec1 2D	35,88	1,0594	6,19	54,44	34,09%
adaptec1 3D	36,37	1,0739	17,36	88,45	58,88%
adaptec2 2D	33,21	1,0371	6,36	52,30	36,50%
adaptec2 3D	33,74	1,0536	18,72	89,89	62,47%
adaptec3 2D	96,09	1,0295	11,60	130,89	26,59%
adaptec3 3D	97,02	1,0395	34,21	199,66	51,41%
adaptec4 2D	90,02	1,0143	11,66	125,00	27,98%
adaptec4 3D	91,28	1,0285	30,56	182,96	50,11%
adaptec5 2D	102,79	1,0499	16,45	152,13	32,43%
adaptec5 3D	103,89	1,0612	52,03	259,98	60,04%
newblue1 2D	24,15	1,0400	7,76	47,42	49,07%
newblue1 3D	24,15	1,0400	23,37	94,26	74,38%
newblue2 2D	46,81	1,0179	9,90	76,51	38,82%
newblue2 3D	47,91	1,0418	28,08	132,16	63,75%
newblue3 2D	75,63	1,0253	11,20	109,23	30,76%
newblue3 3D	75,63	1,0253	32,69	173,71	56,46%

Tabela 3.3: Comparação do roteamento 2D com *layer assignment* e roteamento 3D completo para os *benchmarks* do ISPD 2007.

Bench-mark	*Layer Assignment*					Roteamento 3D completo				
	Ovfl total	WL (e5)	Vias (e5)	Custo total	CPU (min)	*Ovfl* total	WL (e5)	Vias (e5)	Custo total	CPU (min)
a1	0	36,37	17,36	**88,45**	430	1456	36,02	17,55	88,70	1453
a2	0	33,74	18,71	**89,89**	64	2	33,36	19,06	90,54	1444
a3	0	97,02	34,21	**199,66**	243	2	96,69	34,77	201,01	1487
a4	0	91,28	30,56	182,96	55	0	91,39	29,32	**179,36**	83
a5	0	103,89	52,03	**259,98**	740	5512	102,78	52,27	259,61	1462
n1	514	24,15	23,37	**94,26**	1442	1012	24,21	22,33	91,19	1447
n2	0	47,91	28,08	132,16	10	0	47,93	27,15	**129,40**	18
n3	39828	75,63	32,69	**173,71**	1501	51098	75,73	29,30	163,63	1827

Tabela 3.4: Resultados de *overflow* e comprimento de fio do FGR para todos os circuitos do ISPD 1998.

Benchmark	BoxRouter 1.0		FastRoute 2.0		FGR 1.0		vs. Box-Router 1.0	vs. Fast-Route 2.0
	Ovfl	WL	*Ovfl*	WL	*Ovfl*	WL		
ibm1	102	65588	31	68489	0	**63332**	-3,44%	-7,53%
ibm2	33	178759	0	178868	0	**168918**	-5,51%	-5,56%
ibm3	0	151299	0	150393	0	**146412**	-3,23%	-2,65%
ibm4	309	173289	64	175037	0	**167101**	-3,57%	-4,53%
ibm5	0	409747	-	-	0	**409739**	-0,00%	-
ibm6	0	282325	0	284935	0	**277608**	-1,67%	-2,57%
ibm7	53	378876	0	375185	0	**366180**	-3,35%	-2,40%
ibm8	0	415025	0	411703	0	**404714**	-2,48%	-1,70%
ibm9	0	418615	3	424949	0	**413053**	-1,33%	-2,80%
ibm10	0	593186	0	595622	0	**578795**	-2,43%	-2,83%
Média							-2,71%	-3,64%

no comprimento total de fio.

Esse efeito sugere que o roteamento 2D tradicional pode ser mais bem sucedido que o roteamento com grafo 3D, já que neste último as arestas teriam custos menores, provocando uma discretização maior como a que ocorre no problema do "último suspiro". Isso fica claro nos resultados da Tabela 3.3, onde o tempo de execução do roteamento 3D é muito maior que para roteamento 2D com assinalamento de camadas (*layer assignment*) sendo possivelmente a causa desse aumento no tempo de execução a maior complexidade do roteamento 3D.

Os resultados de *overflow* e comprimento de fio para os *benchmarks* do ISPD 1998 e do ISPD 2007 são mostrados na Tabela 3.4 e na Tabela 3.5, respectivamente.

O FGR consegue completar o roteamento de todos os circuitos do ISPD 1998, diferentemente de todos os roteadores anteriores, sendo mostrado na Tabela 3.4 o comparativo com os roteadores BoxRouter 1.0 (CHO; PAN, 2007) e FastRoute 2.0 (PAN; CHU, 2007). Já na Tabela 3.5 estão os resultados para a versão 1.1 do FGR para os *benchmarks* do ISPD 2007. Neste caso, o FGR supera o BoxRouter em comprimento de fio em 10.6%, o MaizeRouter em 9,1%, o Archer (OZDAL; WONG, 2007) em 10,1% e o BoxRouter 2.0 (CHO et al., 2007) em 4,9%.

Tabela 3.5: Resultados de *overflow* e comprimento de fio do FGR para todos os circuitos do ISPD 2007.

Benchmark	Melhor entre BoxRouter e MaizeRouter				FGR 1.1			
	Overflow		Custo	Roteador	*Overflow*		Custo	vs.
	total	máx.	(e5)		total	máx.	(e5)	Melhor
a1 2D	0	0	58,84	Box	0	0	53,71	-8,72%
a1 3D	0	0	99,61	Maize	0	0	88,02	-11,64%
a2 2D	0	0	55,69	Box	0	0	51,86	-6,88%
a2 3D	0	0	98,12	Maize	0	0	89,96	-8,32%
a3 2D	0	0	137,75	Maize	0	0	130,30	-5,41%
a3 3D	0	0	214,08	Maize	0	0	200,14	-6,51%
a4 2D	0	0	128,45	Maize	0	0	123,97	-3,49%
a4 3D	0	0	194,38	Maize	0	0	178,90	-7,96%
a5 2D	0	0	164,32	Box	0	0	151,47	-7,82%
a5 3D	0	0	298,08	Box	0	0	260,53	-12,60%
n1 2D	400	2	51,13	Box	234	2	46,42	-9,21%
n1 3D	400	2	101,83	Box	238	2	90,68	-10,95%
n2 2D	0	0	79,64	Maize	0	0	75,78	-4,85%
n2 3D	0	0	139,66	Maize	0	0	129,30	-7,42%
n3 2D	32588	1236	114,63	Maize	38386	1196	107,28	-6,41%
n3 3D	32840	1058	184,40	Maize	38398	400	163,41	-11,38%
Média								-8,13%

3.1.2 BoxRouter

O BoxRouter (CHO; PAN, 2006, 2007; CHO et al., 2007, 2009) é um roteador global dirigido a roteabilidade que usa programação linear inteira (ILP) progressiva juntamente com *maze routing* adaptativo, com uma etapa de pós-roteamento (posteriormente substituída por técnicas mais eficientes). Neste trabalho é proposta uma abordagem diferente para o uso de ILP do que na referência anterior (ALBRECHT, 2001), tornando a técnica mais escalável.

Em sua primeira versão, o BoxRouter seguia o fluxo mostrado na Figura 3.4, sendo que a principal diferença para a versão 2.0 está nas fases após a etapa de *BoxRouting*, onde a etapa de pós-roteamento é substituída por técnicas de redução de congestionamento e *layer assignment* mais eficientes e sistemáticas, como mostrado no fluxo da Figura 3.5.

A abordagem tradicional de ILP para roteamento global (mostrada na Figura 3.6) tem como objetivo minimizar o congestionamento máximo (formulação mín-máx), tentando atingir uma distribuição mais uniforme do congestionamento. Como a formulação sempre inclui no mínimo uma solução de roteamento candidata para cada fio, o roteamento é completo, ou seja, não é necessário nenhum passo adicional (como *maze routing*), a não ser que ainda existam arestas com *overflow*. A Figura 3.7 mostra esta formulação para o problema do exemplo da Figura 3.8. O resultado deste exemplo para ILP tradicional é o mostrado na Figura 3.8 (c), pois é a solução com menor congestionamento máximo.

A principal limitação desta abordagem é o fato que quando há alguma aresta com *overflow* (C maior que a máxima capacidade das arestas) haverá uma explosão do número das arestas com *overflow*, já que por definição o objetivo é minimizar o máximo *overflow*. Isso dificulta a aplicação para circuitos com regiões de grande congestionamento, como é o caso dos *benchmarks* dos concursos do ISPD. Outro problema desta abordagem tra-

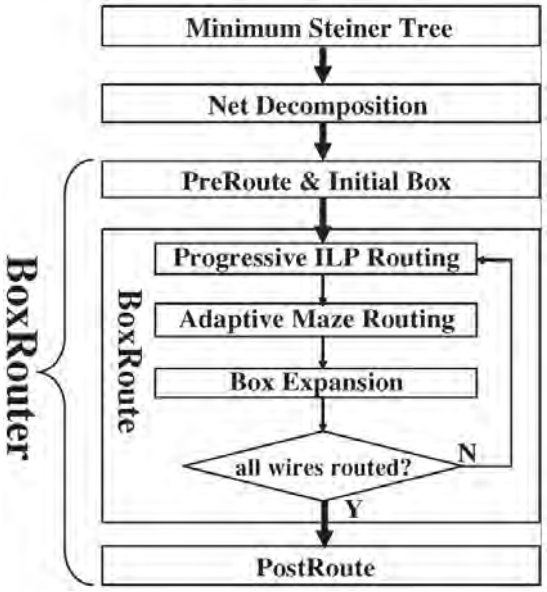

Figura 3.4: Fluxo de execução do BoxRouter. (CHO; PAN, 2007).

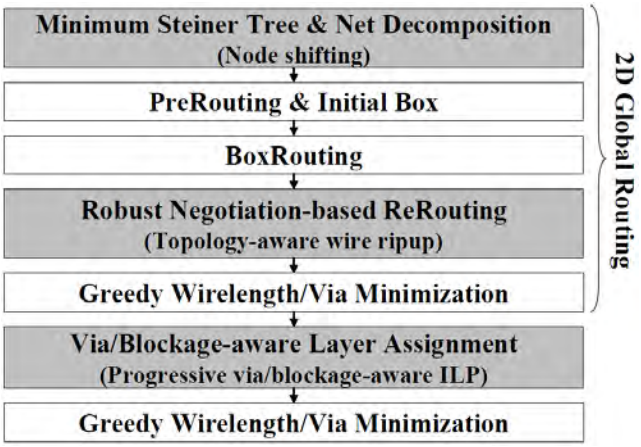

Figura 3.5: Fluxo de execução do BoxRouter 2.0. (CHO et al., 2007).

dicional é que ela não é eficientemente resolvida por algoritmos *branch-and-bound* ou *branch-and-cut*, que são os mais usados em resolvedores ILP.

Assim, a nova formulação proposta por CHO; PAN (2007) considera pesos para os roteamentos candidatos (o que facilita a incorporação de outros objetivos de roteamento, como desempenho), tendo como objetivo maximizar a quantidade de fio roteados, não limitando o roteamento pelo congestionamento máximo mas sim pela capacidade de congestionamento das arestas, sem causar excesso de congestionamento nas arestas e pode ser resolvida eficientemente por algoritmos *branch-and-bound* ou *branch-and-cut*. A formulação proposta está na Figura 3.9.

As principais limitações dessa nova formulação são: a dependência do resolvedor, pois, como a distribuição do congestionamento é ignorada, existem várias possibilidades de solução válida, sendo que esta será escolhida pelo resolvedor, sem o critério da uniformidade do congestionamento, e o fato de necessitar de um passo adicional para completar o roteamento, já que as restrições não garantem que todos os fios serão roteados.

Para o exemplo da Figura 3.8 (b) a nova formulação proposta poderia resultar nas

$$
\begin{array}{lll}
\textbf{minimizar:} & C & \\
\textbf{desde que:} & x_{ijk} \in \{0,1\} & \forall (i,j,k) \in N \\
& \sum_{k:(i,j,k)\in N} x_{ijk} = 1 & \forall i,j \\
& \sum_{(i,j,k)\in L(e)} x_{ijk} \leq C & \forall e
\end{array}
$$

Figura 3.6: Formulação geral da ILP tradicional. Adaptada de (CHO; PAN, 2007).

$$
\begin{array}{ll}
\textbf{minimizar:} & C \\
\textbf{desde que:} & x_{a11}, x_{a12}, x_{a21}, x_{a31}, x_{b11}, x_{b12} \in \{0,1\} \\
& x_{a11} + x_{a12} = 1 \\
& x_{b11} + x_{b12} = 1 \\
& x_{a21} = 1, x_{a31} = 1 \\
& x_{a11} + x_{b12} \leq C \\
& x_{a21} + x_{b11} \leq C \\
& x_{a11} \leq C, x_{a12} \leq C, x_{a31} \leq C \\
& x_{b11} \leq C, x_{b12} \leq C
\end{array}
$$

Figura 3.7: Exemplo de aplicação da ILP tradicional para a Figura 3.8 (b) . Adaptada de (CHO; PAN, 2007).

soluções da Figura 3.8 (c) ou (d), já que não minimiza o congestionamento máximo. A formulação para esta exemplo é mostrada na Figura 3.10.

A técnica de ILP é então aplicada inicialmente para a região (*box*) que cobre a área mais congestionada, sendo progressivamente expandida até cobrir o circuito inteiro. Em cada expansão o *BoxRouter* divide o circuito em duas partes: dentro do *box* e fora do *box*, sobre as quais são aplicadas diferentes estratégias para maximizar a roteabilidade e minimizar o roteamento.

Dentro do *box* o roteamento usa todos os recursos possíveis (abordagem gulosa - *greedy*), já que os fios dentro do *box* têm prioridade em relação aos que estão fora do *box*. Fora do *box* o uso da capacidade de roteamento é mais conservador, já que os fios fora do *box* podem precisar desses recursos posteriormente para viabilizar seu roteamento. A aplicação dessas duas técnicas mantém o circuito denso, como mostrado na Figura 3.11.

A primeira fase do fluxo do BoxRouter é a decomposição das redes em fios de dois pinos, como mostrado na Figura 3.12, através da construção por RSMT, usando o FLUTE, sendo que qualquer algoritmo diferente (dirigidos a desempenho ou congestionamento) para a construção destas redes pode ser usado. Após isso, cada fio da rede se torna um único objeto de roteamento, o que pode resultar em um roteamento sub-ótimo, já que há perda de informações sobre os outros fios da rede. Este problema é contornado pelo uso do *maze routing* adaptativo (AMR).

A fase de pré-roteamento faz o roteamento do máximo possível de "fios planos"(aqueles que não possuem curvas, como os fios $a-e, e-d, e-f$ e $b-f$ da Figura 3.12) através do menor caminho, desde que não cause nenhum *overflow*. Esta etapa traz grande melhoria no tempo de execução sem causar degradação na solução final. De acordo com os experimentos realizados, cerca de 60% do comprimento de fio total pode ser roteado com

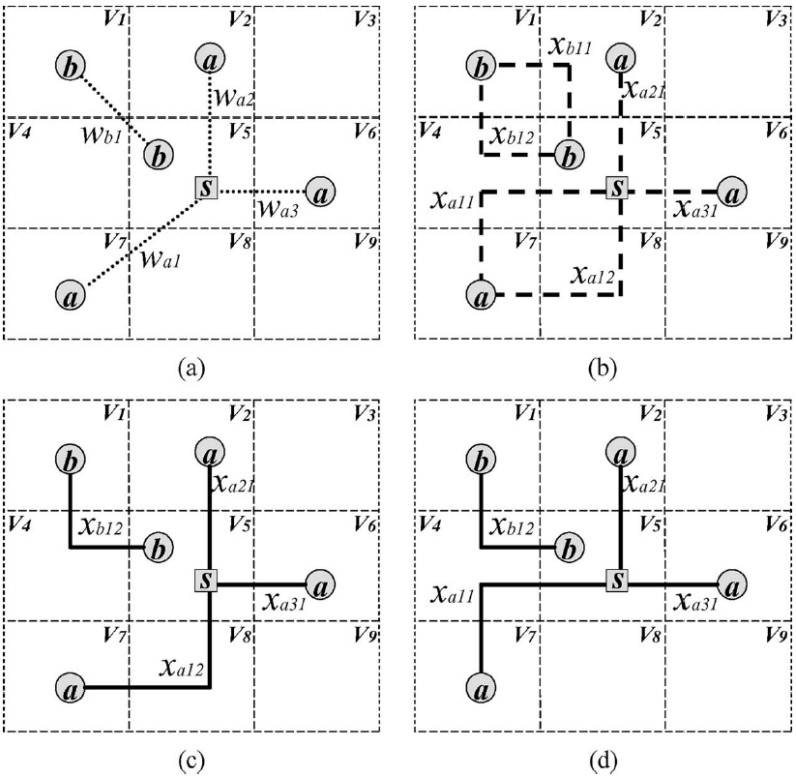

Figura 3.8: Exemplo de uso de ILP em roteamento global. (CHO; PAN, 2007).

$$
\begin{array}{lll}
\textbf{maximizar:} & \sum_{(i,j,k)\in N} a_{ijk} \cdot x_{ijk} & \\
\textbf{desde que:} & x_{ijk} \in \{0,1\} & \forall (i,j,k) \in N \\
& \sum_{k:(i,j,k)\in N} x_{ijk} \leq 1 & \forall i,j \\
& \sum_{(i,j,k)\in L(e)} x_{ijk} \leq c_e & \forall e
\end{array}
$$

Figura 3.9: Formulação geral da ILP proposta. Adaptada de (CHO; PAN, 2007).

$$
\begin{array}{ll}
\textbf{maximizar:} & 2x_{a11} + 2x_{a12} + x_{a21} + x_{a31} + 2x_{b11} + 2x_{b12} \\
\textbf{desde que:} & x_{a11}, x_{a12}, x_{a21}, x_{a31}, x_{b11}, x_{b12} \in \{0,1\} \\
& x_{a11} + x_{a12} \leq 1 \\
& x_{b11} + x_{b12} \leq 1 \\
& x_{a21} \leq 1, x_{a31} \leq 1 \\
& x_{a11} + x_{b12} \leq 2 \\
& x_{a21} + x_{b11} \leq 2 \\
& x_{a11} \leq 2, x_{a12} \leq 2, x_{a31} \leq 2 \\
& x_{b11} \leq 2, x_{b12} \leq 2
\end{array}
$$

Figura 3.10: Exemplo de aplicação da ILP proposta para a Figura 3.8 (b) . Adaptada de (CHO; PAN, 2007).

tempo de execução irrisório pela fase de pré-roteamento.

O congestionamento da fase de pré-roteamento gera uma boa estimativa dos pontos mais congestionados, como mostrado na Figura 3.13 e serve como referência para a defi-

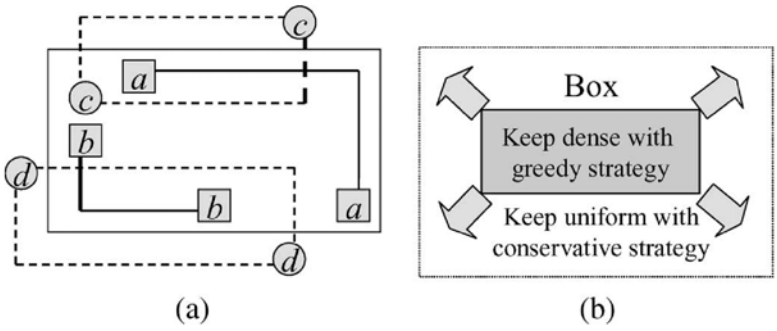

Figura 3.11: Motivação (a) e estratégias (b) para *BoxRouting*. (CHO; PAN, 2007).

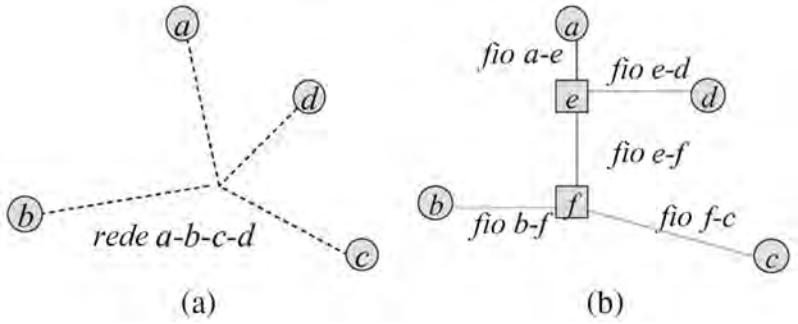

Figura 3.12: Decomposição de uma rede (a) em fios de dois pinos (b) através de RSMT. Adaptada de (CHO; PAN, 2007).

nição do *box* inicial. Esse *box* engloba as quatro GRCs na área mais congestionada, como mostrado na Figura 3.14 (a). Caso existam duas ou mais regiões mais congestionadas, a mais próxima do centro do circuito é escolhida.

A Figura 3.14 mostra um exemplo de *BoxRouting*. A partir da escolha da região mais congestionada após o pré-roteamento (3.14 (a)), considera-se o *box* i e então classifica-se os fios em dentro do *box* (fios com pontos quadrados - b, f e h) ou fora do *box* (fios com pontos circulares - a, c, d, i e k). Os fios já roteados pela fase anterior são mostrados com linhas sólidas e os fios a serem roteados usando ILP são mostrados na Figura 3.14 (c) em suas respectivas GRCs (v_A, v_B, v_C e v_D), sendo a formulação ILP mostrada na Figura

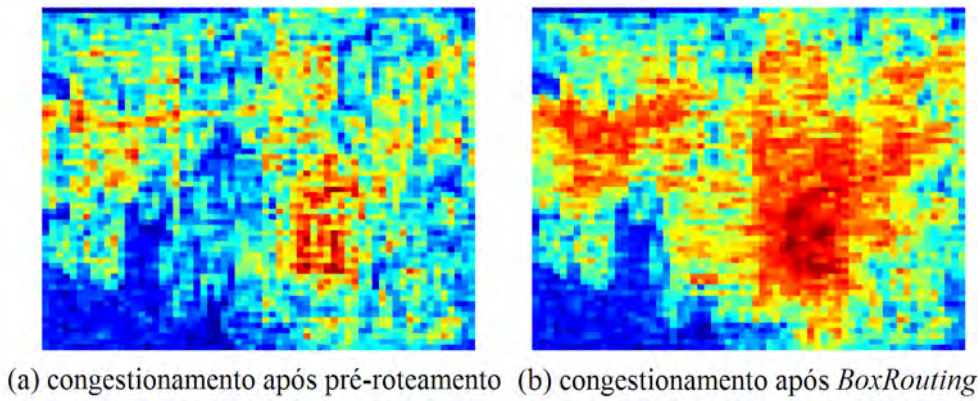

Figura 3.13: Estimativa de roteamento após o (a) pré-roteamento e após o (b) *BoxRouting*. Adaptada de (CHO; PAN, 2007).

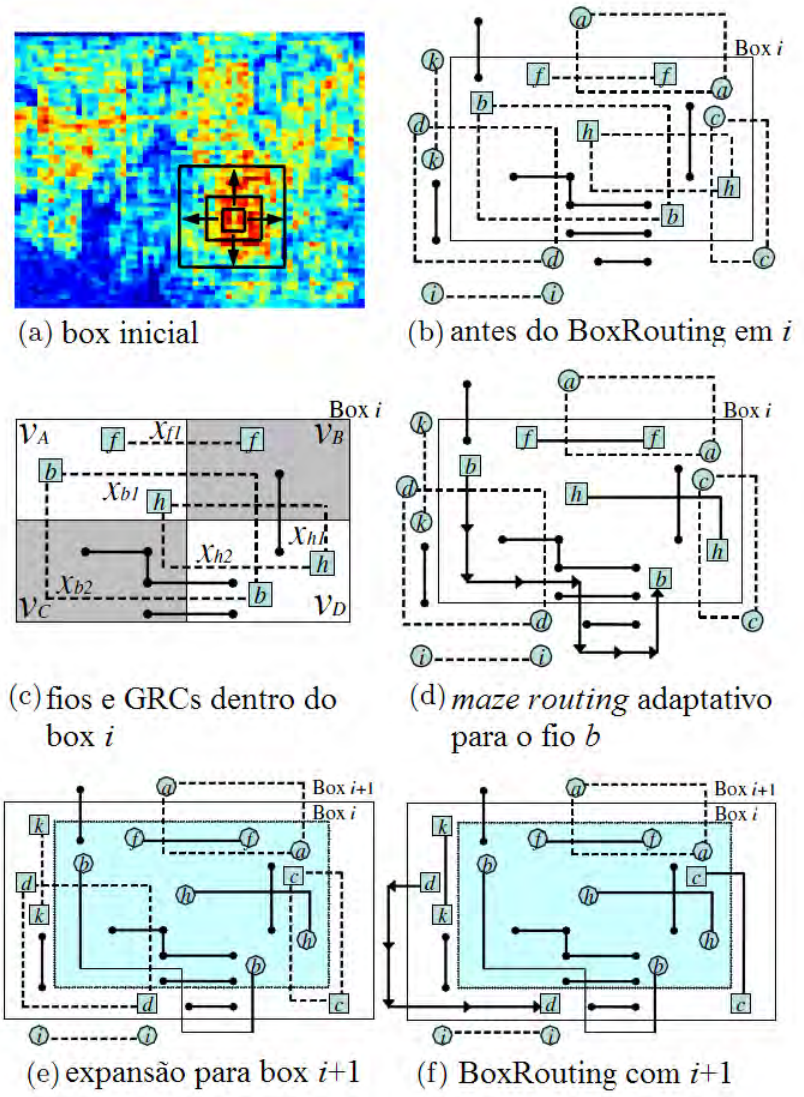

Figura 3.14: Exemplo de *BoxRouting*. (a) *Box* inicial na área mais congestionada. (b) *Box* i com fios internos e externos. (c) Fios internos que serão roteados pela ILP. (d) Fio b roteado com AMR. (e) *Box* expandido englobando mais fios. (f) *BoxRouting* executado novamente para o *Box* $i + 1$. (CHO; PAN, 2007).

3.15. Para fios "planos"apenas um candidato ao roteamento é considerado ($x_{f2} = 0$), sendo que para os demais são consideradas duas formas L como candidatas (x_{b1}, x_{b2}, x_{h1} e x_{h2}). O problema é então solucionado pelo resolvedor ILP.

Como nem todos os fios podem ser roteados com formas L ou linhas retas, os fios que não foram roteados pela ILP (b no exemplo) são então roteados usando AMR, cuja diferença para o *maze routing* normal está na função custo. Para arestas dentro do *box* o custo é sempre unitário, desde que ainda haja capacidade de roteamento na aresta, e para as arestas fora do *box* o custo é a quantidade de recursos já utilizados (diferença entre máximo e disponível). O método proposto é implementado usando *maze routing* para múltiplas origens e múltiplos destinos com ponte (MMB), onde caminhos já existentes na rede do fio a ser roteado são considerados para buscar menor comprimento de fio, como mostrado no exemplo da Figura 3.16.

Após isso o *box* será expandido para o *box* $i + 1$ (Figura 3.14 (e)), englobando mais

$$\textbf{maximizar: } x_{b1} + x_{b2} + x_{f1} + x_{f2} + x_{h1} + x_{h2}$$
$$\textbf{desde que: } x_{b1}, x_{b2}, x_{f1}, x_{f2}, x_{h1}, x_{h2} \in \{0, 1\}$$
$$x_{b1} + x_{b2} \leq 1$$
$$x_{f1} + x_{f2} \leq 1$$
$$x_{f2} = 0$$
$$x_{h1} + x_{h2} \leq 1$$
$$x_{b1} + x_{f1} + x_{h1} \leq c_{AB}$$
$$x_{b1} + x_{h1} \leq c_{BD}$$
$$x_{b2} + x_{h2} \leq c_{AC}$$
$$x_{b2} + x_{h2} \leq c_{CD}$$

Figura 3.15: Formulação ILP para o problema da Figura 3.14 (c). Adaptada de (CHO; PAN, 2007).

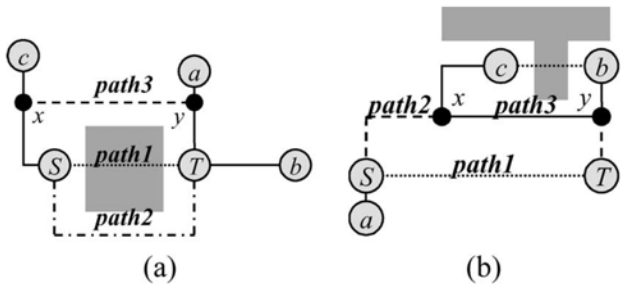

Figura 3.16: Exemplo de AMR com MMB. (a) caminho *path3* é encontrado para evitar desvio do caminho *path2*. (b) compartilhamento do caminho *path3* já roteado diminui o comprimento de fio total. (CHO; PAN, 2007).

fios (c, d e k) e repetindo o processo. O tamanho do incremento afeta consideravelmente a qualidade do roteamento e o tempo de execução da ferramenta. Expansões maiores provocam menos *overflow* (já que tratam o problema mais globalmente) mas causam um aumento exponencial do tempo de execução do resolvedor ILP, como mostrado no gráfico da Figura 3.17.

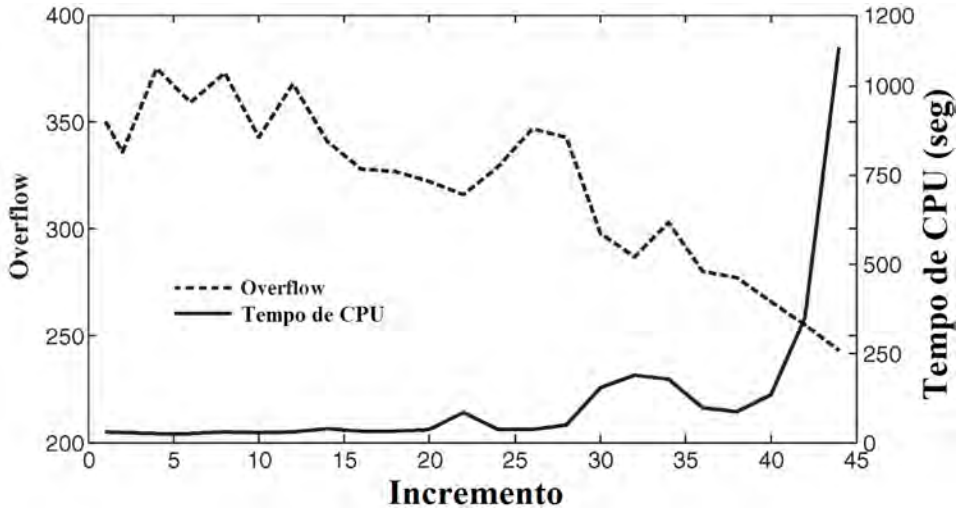

Figura 3.17: *Overflow* e tempo de execução vs. tamanho do incremento do *box*. Adaptada de (CHO; PAN, 2007).

A fase seguinte, que difere as duas versões do BoxRouter, faz o roteamento final das redes que ainda não possuem um roteamento válido. Isso é feito com o uso de pesquisa *A** baseada em negociação, com a função custo definida abaixo:

$$cost^i(e) \quad = h^i(e) + \alpha p(e) + \beta d(e)$$

onde, para uma aresta e, $h^i(e)$ é o histórico do custo na i-ésima iteração, $p(e)$ a utilização atual da aresta e $d(e)$ a distância de e para o destino.

Para problemas com muito congestionamento a pesquisa *A** apresenta problemas de estabilidade, já que o valor do histórico ($h^i(e)$) passa a dominar o custo presente ($p(e)$), fazendo com que arestas congestionadas no momento sejam mais "baratas"que as anteriormente muito congestionadas. Para contornar este problema foi introduzido o fator de escala α, definido abaixo:

$$\alpha = \frac{max_e[h^i(e)]}{p(e)|_{1.0}}$$

onde $p(e)|_{1.0}$ indica o custo de congestionamento quando não há nenhuma capacidade de roteamento disponível na aresta e.

Esta técnica é aplicada juntamente com uma técnica de *rip-up and reroute* modificada para que não só o fio passando por uma região com *overflow* seja refeito mas também outros fios próximos pertencentes a mesma rede, o que proporciona maior flexibilidade para a pesquisa *A**.

A fase final do BoxRouter é o assinalamento dos camadas que também é feito usando uma formulação para ILP, buscando a minimização das vias.

Tabela 3.6: Resultados de *overflow* e comprimento de fio do BoxRouter 2.0 para todos os circuitos do ISPD 2007.

Benchmark	WL (e5)	Máx. *Ovfl*	*Ovlf*
adaptec1 2D	58,37	0	0
adaptec1 3D	92,04	0	0
adaptec2 2D	55,69	0	0
adaptec2 3D	94,28	0	0
adaptec3 2D	137,96	0	0
adaptec3 3D	207,41	0	0
adaptec4 2D	127,79	0	0
adaptec4 3D	186,42	0	0
adaptec5 2D	162,11	0	0
adaptec5 3D	270,41	0	0
newblue1 2D	51,13	2	400
newblue1 3D	92,94	2	394
newblue2 2D	78,68	0	0
newblue2 3D	134,64	0	0
newblue3 2D	111,61	1088	38958
newblue3 3D	172,44	364	38958

Os resultados do BoxRouter 2.0 para os *benchmarks* do ISPD 2007 são mostrados na Tabela 3.6.

3.1.3 MaizeRouter

O MaizeRouter (MOFFITT, 2008b,a), vencedor da categoria 3D do ISPD *Routing Contest* 2007, baseia-se basicamente em duas técnicas: *extreme edge shifting* e *edge retraction*. O fluxo de execução do MaizeRouter pode ser visto na Figura 3.18. Como outros roteadores já citados, o MaizeRouter faz o mapeamento de circuitos 3D para problemas 2D.

A geração das RSMTs é feita usando o pacote FLUTE e a sua decomposição em redes de dois terminais é feita de modo a agrupar os segmentos de uma mesma rede, permitindo futuras mudanças na topologia da rede.

Sobre essa solução inicial é então aplicada a técnica de deslocamento de arestas (*edge shifting*), que permite o aumento do comprimento de fio para que a rede evite passar por caminhos congestionados, como mostrado no exemplo da Figura 3.19 (a) e (b), onde a rede como um todo é considerada (sem haver a construção em árvore), para evitar que existam caminhos repetidos, mudando-se, portanto, a topologia da rede em certos casos, sem que haja o *rip-up* dos segmentos. Este passo é repetido para todos os fios do circuito.

Após isso segue-se o passo de desfragmentação das redes e retração das arestas. A desfragmentação das arestas une os fios (de dois terminais) de uma mesma rede que estejam na mesma direção, ou seja, que na prática representem apenas um fio. Esta técnica dá mais liberdade para a atuação da retração das arestas, que faz deslocamentos das arestas limitado pela região dos demais fio da mesma rede, o que evita o aumento do comprimento de fio. Este processo é mostrado na Figura 3.20. A necessidade de desfragmentação das redes é exemplificada na Figura 3.21.

Estes dois passos têm agregados a si o *garbage collection*, que faz a remoção das arestas que são desnecessárias após a aplicação de cada técnica. O *garbage collection*

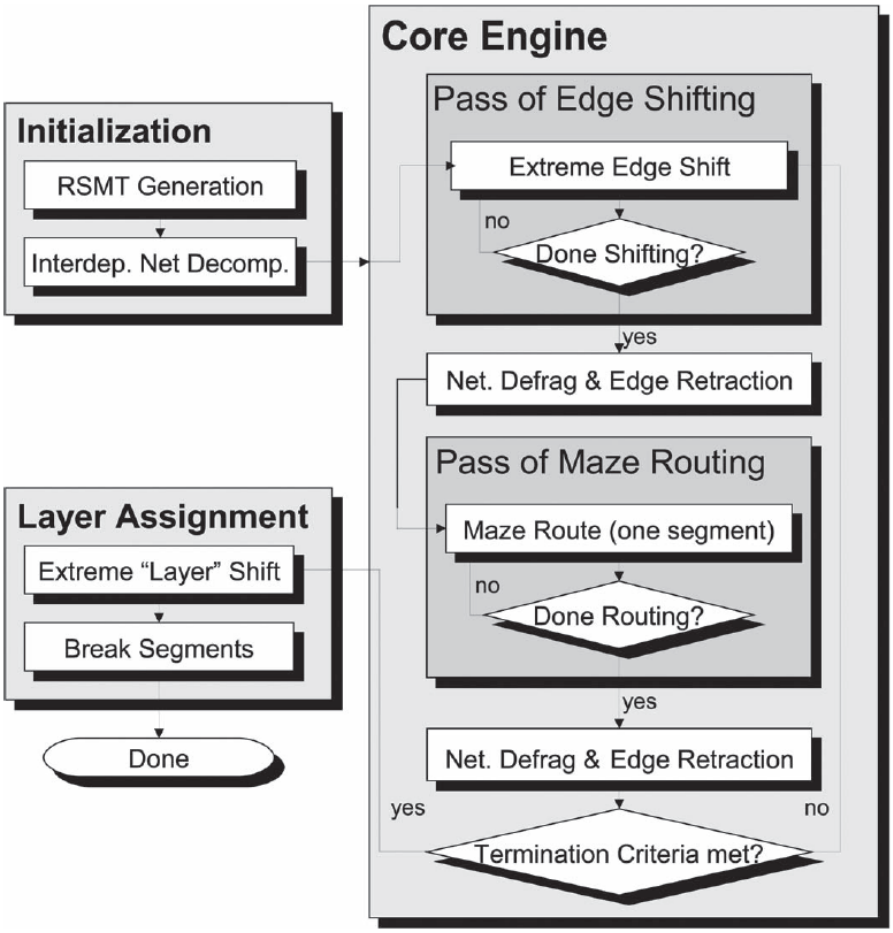

Figura 3.18: Fluxo de execução do MaizeRouter. (MOFFITT, 2008a).

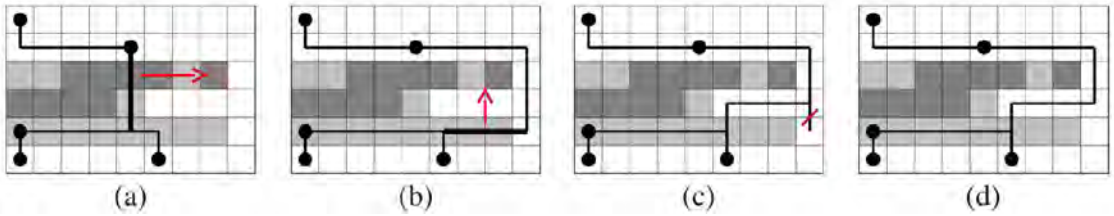

Figura 3.19: Exemplo de *edge shifting* e *garbage collection*. (MOFFITT, 2008a).

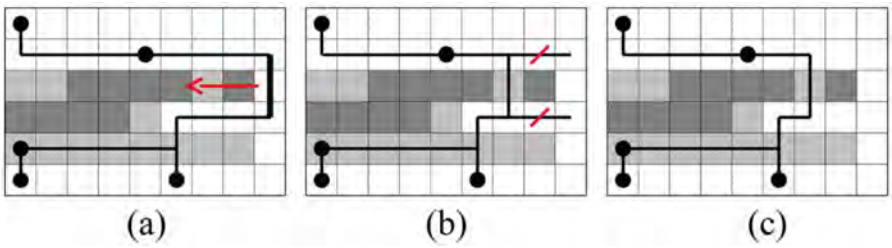

Figura 3.20: Exemplo de *edge retraction* e *garbage collection*. (MOFFITT, 2008a).

Tabela 3.7: Resultados de *overflow* e comprimento de fio do MaizeRouter para todos os circuitos do ISPD 2007.

Benchmark	WL (e5)	*Ovfl*	CPU (min)
adaptec1 2D	55,91	0	221
adaptec1 3D	93,79	0	223
adaptec2 2D	53,22	0	40
adaptec2 3D	91,65	0	42
adaptec3 2D	133,49	0	80
adaptec3 3D	202,13	0	83
adaptec4 2D	124,95	0	13
adaptec4 3D	185,68	0	14
adaptec5 2D	157,50	0	337
adaptec5 3D	265,90	0	339
newblue1 2D	46,97	1194	84
newblue1 3D	95,27	1194	87
newblue2 2D	76,48	0	7
newblue2 3D	133,48	0	9
newblue3 2D	110,56	32825	180
newblue3 3D	177,44	32825	182

também é mostrado na Figura 3.19 e na Figura 3.20.

Após isso, a ferramenta utiliza o *maze routing* para rotear as redes que ainda não possuem caminhos válidos (aquelas cujos desvios são mais complexos do que os tratados pelo deslocamento de arestas). A função custo do *maze routing* utilizada pelo MaizeRouter apresenta uma modificação para que os custos das arestas não sejam representados por um salto (custo zero para arestas sem *overflow* e um para arestas com *overflow*) e nem por um custo linear (diferença entre a capacidade de roteamento e recursos já utilizados) durante toda a execução do algoritmo. Para isso é usada a função custo abaixo:

$$\text{custo} = 1 + \frac{h}{1 + e^{-k(\text{demanda - capacidade})}} - h$$

onde h controla o compromisso entre o congestionamento e o comprimento de fio (valor zero o custo depende apenas do comprimento de fio e valor um depende apenas do congestionamento) e k controla a suavidade da curva. Esta função custo é atualizada dinamicamente, sendo os valores dessas duas variáveis aumentados a cada iteração, para que ao final do processo o custo dependa apenas do congestionamento (função salto), como mostrado na Figura 3.22

Estes passos são repetidos até que o critério de parada estabelecido seja alcançado, por exemplo, que o roteamento esteja completo ou um tempo de execução limite seja alcançado. Para a geração da solução final o problema é então mapeado para várias camadas de metal (3D) usando a mesma técnica de deslocamento de arestas que a usada para tratar o congestionamento.

Os resultados do MaizeRouter para os circuitos do ISPD 2007 são mostrados na Tabela 3.7.

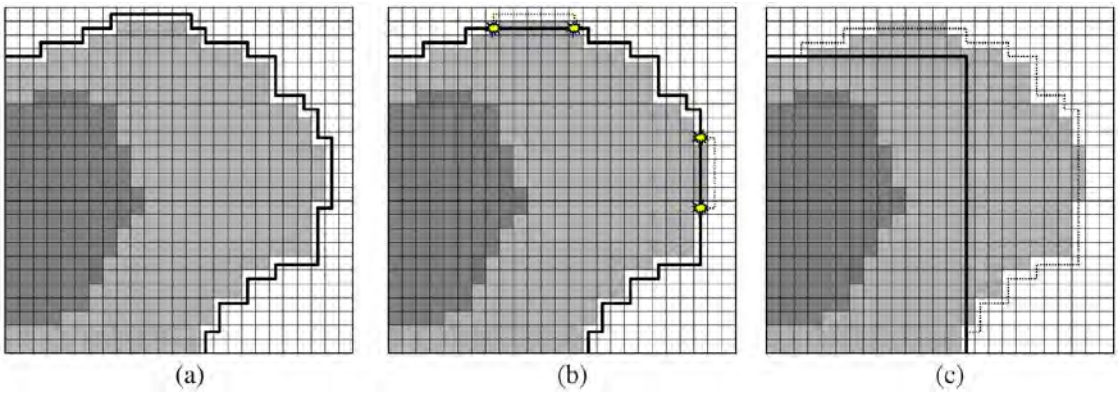

Figura 3.21: (a) Desvios excessivos causados pelo *maze routing* para evitar regiões perto da capacidade máxima. (b) aplicação da retração de aresta sozinha. (c) resultado final após a aplicação repetida de retração de aresta e desfragmentação de rede. (MOFFITT, 2008a).

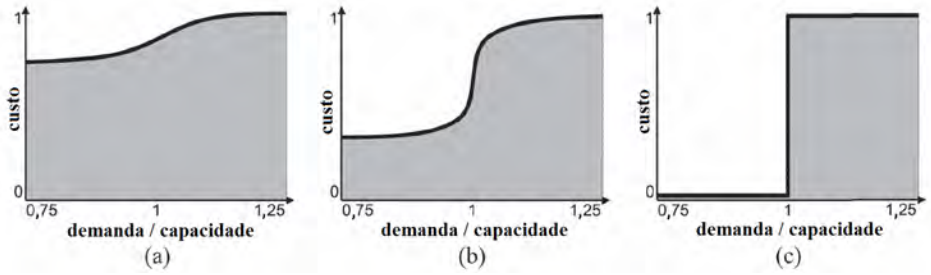

Figura 3.22: Função custo dinâmica ao longo da execução do processo. Adaptada de (MOFFITT, 2008a).

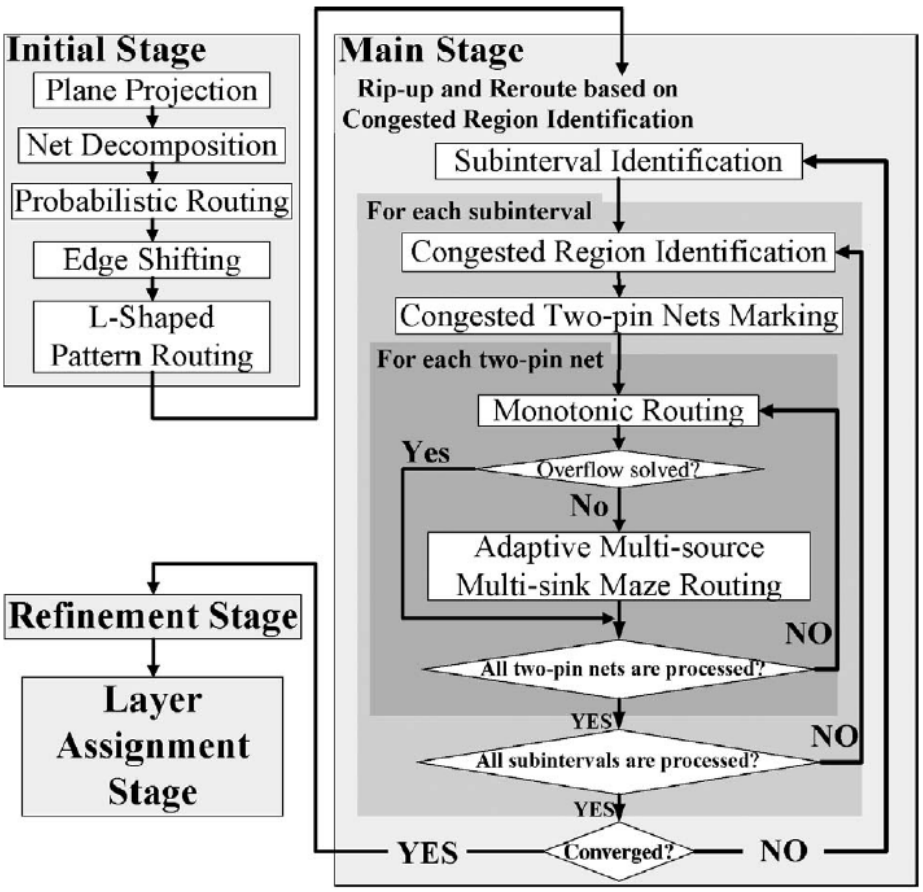

Figura 3.23: Fluxo de execução do NTHU-Route 1.0. (CHANG et al., 2010).

3.2 ISPD *Contest* 2008

Neste concurso foram usados apenas os circuitos 3D do concurso anterior e foram introduzidos oito novos *benchmarks*. A avaliação do comprimento de fio leva em consideração o tempo de execução das ferramentas, conforme a equação abaixo:

$$WL_{wt} = WL \times (1 + min(0,1,0,04 \times log_2(\frac{tempo_{cpu}}{tempo\_medio_{cpu}})))$$

onde WL é a medida do ISPD 2007, com peso de via unitário, $tempo_{cpu}$ é o tempo de execução do roteador e $tempo\_medio_{cpu}$ é o tempo médio de execução de todos os roteadores. Essa métrica permite uma redução de 0,4% no comprimento de fio caso o roteador seja duas vezes mais rápido que a média, limitados a 10% de redução.

3.2.1 NTHU-Route

O NTHU-Route (GAO; WU; WANG, 2008; CHANG; LEE; WANG, 2008; CHANG et al., 2010; LEE; CHANG; WANG, 2010), vencedor do ISPD *Contest* 2008 com a versão 2.0, é um roteador global baseado em *rip-up and reroute* em conjunto com diversas técnicas de otimização.

Como mostrado no fluxo da Figura 3.23 a primeira versão proposta do NTHU-Route era dividida em quatro fases principais: estágio inicial, estágio principal, refinamento e assinalamento de camadas.

Na fase inicial o roteador projeta as camadas para um plano (de 3D para 2D) e usa o FLUTE para decompor as redes multiterminais em redes de dois terminais. Após isso, as redes são roteadas probabilisticamente usando as duas formas de L possíveis, atribuindo uma demanda de 0,5 para cada uma ou usando demanda 1 para redes em linha reta. A seguir, o roteador modifica a topologia das redes, quando apropriado, usando a técnica de *edge shifting* (PAN; CHU, 2006) e então as redes são novamente roteadas com formas L, escolhendo-se a de menor custo. Em ambas etapas o NTHU-Route usa a função custo de (PAN; CHU, 2006).

No estágio principal, o roteador usa a técnica de *rip-up and reroute* iterativo para melhorar a solução inicial nas regiões onde há *overflow*. Essas regiões congestionadas são identificadas primeiramente sobre a solução inicial, onde são calculados os congestionamentos de todas as arestas e é definido um intervalo entre o maior congestionamento e 1. Esse intervalo é dividido então em m subintervalos $\{I_1, I_2, ..., I_m\}$, sendo m definido como 10 no NTHU-Route. Por exemplo, quando o congestionamento máximo é 2, os subintervalos são: [2, 1,9),[1,9, 1,8),[1,8, 1,7), ...,[1,1, 1). O roteador sequencialmente seleciona cada aresta e cujo valor de congestionamento esteja em I_1, começando pela mais congestionada, e expande uma região retangular r_e em torno de e até que o congestionamento médio da região seja menor que o limite inferior de I_1. Todas as redes de dois pinos que passem por no mínimo uma aresta com *overflow* e que tenham os dois pinos dentro de r_e são marcadas. Após marcar todas as redes o roteador começa a fase de *rip-up and reroute* para uma rede por vez, começando pela que tem o menor perímetro, utilizando roteamento monotônico (PAN; CHU, 2007) ou *maze routing* adaptativo com múltiplas origens e múltiplos destinos (caso o monotônico não encontre um caminho sem *overflow*). Após processar todas as redes de I_1, o roteador aplica as mesmas técnicas para os demais subintervalos até que todos sejam processados. Por fim, o roteador aplica *rip-up and reroute* para as redes que eventualmente não tenham sido marcadas.

O *maze routing* adaptativo com múltiplas origens e múltiplos destinos (*adaptative multi-source and multi-sink maze routing* - AMMMR) considera apenas os pinos e os nodos de Steiner como origens ou destinos para o algoritmo, diminuindo o espaço de busca do algoritmo similar proposto em (PAN; CHU, 2007), que considera todos os pontos da grade de uma subárvore como origens ou destinos.

A função custo baseada em histórico usada no estágio principal é mostrada abaixo:

$$custo_e = b_e + h_e \times p_e + vc_e$$

onde o custo base b_e é setado em 1 (uma unidade de comprimento de fio), $h_e x p_e$ é o custo de congestionamento da aresta e (mostrados a seguir) e vc_eé o custo de via quando se usa a aresta e. A função do histórico h_e é atualizada como descrito abaixo:

$$h_e^{i+1} = \begin{cases} h_e^i + 1 & \text{se } e \text{ possui } overflow \\ h_e^i & \text{caso contrário} \end{cases}$$

onde i é o contador de iterações e $h_e^1 = 1$. O termo de penalização é definido abaixo:

$$p_e = \left(\frac{d_e + 1}{c_e} \right)^{k_1}$$

onde k_1 é um parâmetro definido pelo usuário e setado para 5. O custo de via é mostrado abaixo:

$$vc_e = \begin{cases} 1 & \text{se passar por } e \text{ gera um desvio} \\ 0 & \text{caso contrário} \end{cases}$$

Quando já há uma rede da mesma árvore passando pela mesma aresta, o custo dessa é zerado, permitindo que as redes de dois terminais compartilhem melhor as arestas, reduzindo o comprimento de fio.

A versão 2.0 do NTHU-Route apresenta três modificações (melhorias): mudança na função custo usada durante o estágio principal, mudança no ordenamento das redes a serem re-roteadas e na identificação das regiões congestionadas, e duas técnicas para reduzir o tempo de execução total.

A nova função custo proposta é dividida em três sub-funções: função de custo base, função de custo de congestionamento e função de custo de via. As duas primeiras focam na redução do *overflow* e a última foca na redução do comprimento de fio.

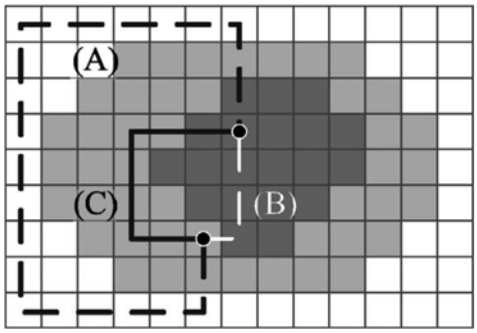

Figura 3.24: Caminhos diferentes para custos base diferentes. (CHANG et al., 2010).

Diferente do que ocorre na primeira versão do roteador, a função de custo base não é mais unitária. O custo unitário faz com que o roteador obtenha caminhos com menor comprimento de fio, em vez de caminhos com menor *overflow*. Entretanto, usar o custo base zero faria com que o roteador ocupasse muitos recursos de roteamento, dificultando o roteamento das outras redes. A Figura 3.24 mostra uma exemplo de diferentes custos base. As regiões mais escuras representam arestas com *overflow*, as intermediárias representam regiões quase com *overflow* e as claras representam arestas com pouca demanda.

O caminho (A) seria o escolhido para um custo base zero (ou próximo a zero), (B) o caminho que não considera o *overflow* (comprimento de fio superestimado) e (C) uma solução intermediária (e melhor). Esses caminhos são todos, de fato, necessários em diferentes casos ao longo da execução do roteador. Durante os passos iniciais do roteamento, caminhos como o (B) são preferíveis, para evitar o uso excessivo de recursos de roteamento que provocariam *overflow* para outras redes. Caminhos como o (C) são preferíveis para evitar excesso de congestionamento nas fases de redução de *overflow*. Já caminhos como (A) são necessários na fase final, onde a geração de uma solução válida (sem *overflow*) tem prioridade sobre o objetivo de menor comprimento de fio total. Para lidar com essas diferenças, foi definida a função custo base abaixo:

$$b_e = 1 - e^{-\beta e^{-\gamma i}}$$

onde β e γ são parâmetros definidos pelo usuário e i o contador de iterações. No NTHU-Route 2.0 os valores são $\beta = 5$ e $\gamma = 0,1$. O comportamento dessa curva é visto na Figura 3.25

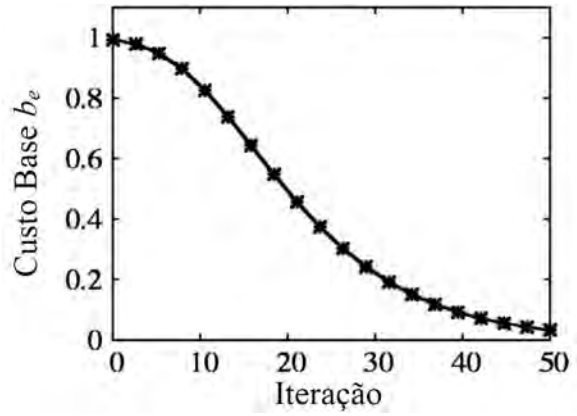

Figura 3.25: Custo base adaptativo. Adaptada de (CHANG et al., 2010).

A função de custo de congestionamento foi adaptada para simular uma predição de congestionamento, fazendo com que o custo de penalização aumente drasticamente mesmo antes que haja *overflow*, considerando o histórico de uso da aresta. Para incluir esse objetivo a equação de p_e foi alterada para a descrita abaixo:

$$p_e = \left(\frac{d_e + 1}{c_e} \times f(h_e, i) \right)^{k_1}$$

onde k_2 é um parâmetro definido pelo usuário e setado para 1,5 como padrão e $f(h_e, i)$ é definida abaixo:

$$f(h_e, i) = \left(\frac{i \times (k_2 + adj(i))}{i \times (k_2 + adj(i)) - (h_e - 1)} \right)$$

sendo

$$adj(i) = k_3 \times (1 - e^{-\beta e^{-\gamma i}})$$

onde k_3 é um parâmetro definido pelo usuário e setado para 3 como padrão.

A Figura 3.26 mostra o comportamento da curva de p_e. O principal efeito desta mudança é a diminuição do número de iterações para uma mesma redução de *overflow*, tornando o roteador mais rápido.

Para adequar o custo das vias para projetos com mais de duas camadas de metal, o novo custo de via foi definido como:

$$vc_e = \begin{cases} v_e \times c_e \times b_e & \text{se passar } e \text{ provoca um desvio} \\ 0 & \text{caso contrário} \end{cases}$$

onde v_e é o número de vias esperado para um desvio, c_e é o custo da via (em unidades de comprimento de fio) e b_e como definido anteriormente.

O novo ordenamento de redes proposto nessa segunda versão do roteador é diferente para a fase de *rip-up and reroute* e para a fase de identificação de regiões congestionadas.

O ordenamento das redes durante a fase de *rip-up and reroute*, como dito anteriormente, era iniciada pela rede com o menor perímetro (*bounding box*), mas como pode ser

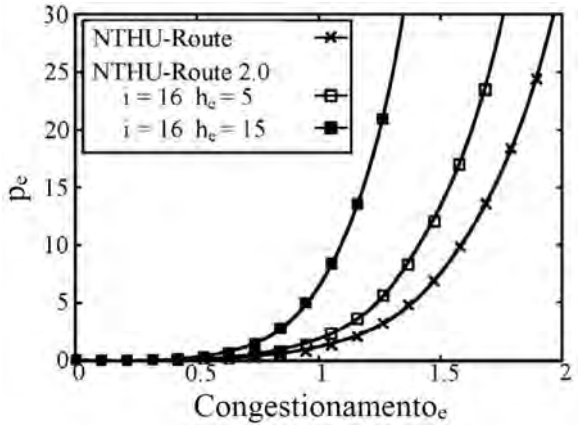

Figura 3.26: Curvas de p_e do NTHU-Route e do NTHU-Route 2.0. Adaptada de (CHANG et al., 2010).

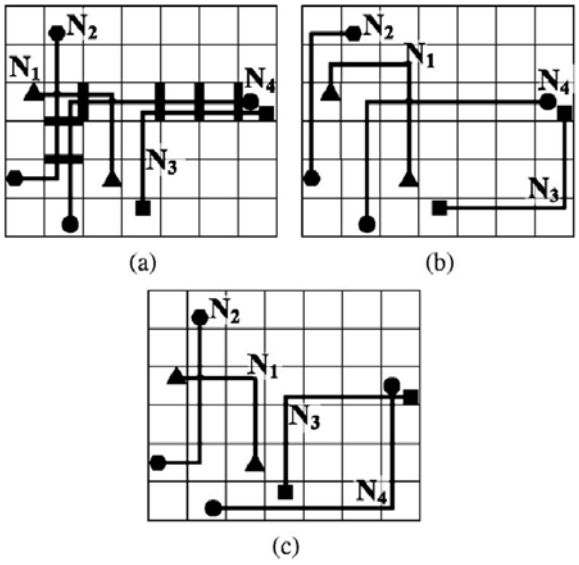

Figura 3.27: Exemplo de diferentes ordenamentos de redes durante a fase de *rip-up and reroute*. (CHANG et al., 2010).

visto pelo exemplo da Figura 3.27, o ordenamento começando pela rede de menor perímetro (b) apresenta maior comprimento de fio que a ordem inversa (c), ou seja, iniciando pela rede de maior perímetro, no caso N_4, a partir de uma iteração anterior (a).

Já o ordenamento durante a fase de identificação das regiões mais congestionadas foi formulado baseado na constatação de que, geralmente, essas regiões têm forma radial, como mostrado na Figura 3.28 (a). Na versão anterior do roteador (Figura 3.28 (b)) o primeiro critério para a escolha das redes a serem re-roteadas era o valor do congestionamento e o segundo o perímetro da rede. Na versão 2.0, além da inversão do critério do perímetro da rede já citado, também há a inversão do critério do valor do congestionamento. Essa mudança foi motivada pelo fato que, iniciando o *rip-up and reroute* pela região r_m do exemplo, estas teriam que fazer grandes desvios para alcançar a região r_p (que não possui *overflow*), causando desperdício de recursos que poderiam ser usados pelas redes em r_n e r_o para evitar *overflow* (Figura 3.28 (c)). Assim, o *rip-up and reroute* começa pela região com menor *overflow* (região r_p do exemplo), permitindo a redução do *overflow* sem que haja um crescimento excessivo do comprimento de fio.

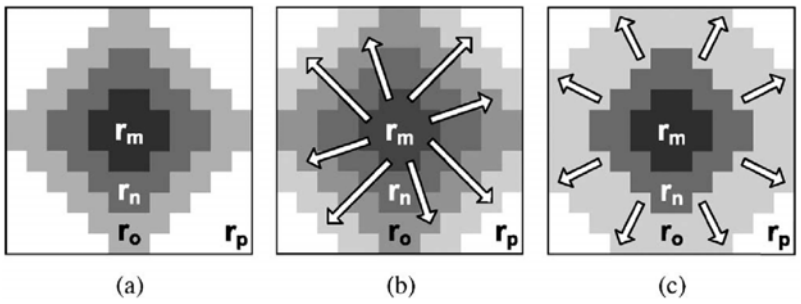

Figura 3.28: Exemplo de diferentes ordenamentos de regiões durante a fase de *rip-up and reroute*. (CHANG et al., 2010).

Tabela 3.8: Resultados de *overflow* e comprimento de fio do NTHU-Route para todos os circuitos do ISPD 1998.

Benchmark	NTHU-Route			NTHU-Route 2.0		
	Overflow total	WL	CPU (s)	*Overflow* total	WL	CPU (s)
ibm01	0	63321	4,3	0	62498	1,54
ibm02	0	170531	7,8	0	169881	3,15
ibm03	0	146551	6	0	146458	1,49
ibm04	0	168262	21,7	0	166452	3,81
ibm06	0	278617	12,4	0	277696	3,16
ibm07	0	366288	16,2	0	366133	4,07
ibm08	0	405169	11	0	404976	3,72
ibm09	0	415464	12,9	0	414738	3,95
ibm10	0	580793	33,6	0	579870	6,99

Por fim, as melhorias no tempo de execução do roteador foram propostas a partir da identificação dos dois principais gargalos do roteador: o cálculo dos custos das arestas de roteamento na busca de caminhos para redes com *overflow* e a identificação se outra parte da rede multiterminal já passa pela mesma aresta. Como o custo de cada aresta não era armazenado na primeira versão do roteador, toda vez que este era necessário fazia-se o cálculo em tempo de execução, o que deixaria o roteador ainda mais lento, já que a função custo da nova versão é mais complexa. Para evitar esse gargalo, o custo base e o custo de congestionamento são calculados e armazenados previamente, sendo atualizados quando necessário. O segundo gargalo é o acesso às redes que já passam por uma aresta. Na primeira versão, as redes eram armazenadas usando uma *lookup table*, sendo que esta foi substituída por uma *hash table*, tornando o acesso mais eficiente. Em média, essas duas técnicas tornaram o NTHU-Route 2,37x mais rápido.

Os resultados do NTHU-Route (duas versões) são mostrados na Tabela 3.8 (ISPD 1998) e na Tabela 3.9 (ISPD 2008, que inclui os circuitos 3D do ISPD 2007)

A versão 3 do NTHU-Route apresenta três modificações para considerar a temperatura do circuito no roteamento. A primeira é na fase inicial, onde as formas L de roteamento são escolhidas de acordo com o número de regiões mais aquecidas (*hot spots*), determinadas por um perfil térmico, pelas quais estas passam, sendo escolhida a com menor número de *hot spots*, ou, em caso de empate, a mesma função custo da versão anterior.

A segunda alteração é nas funções custo durante a fase de *rip-up and reroute*. Durante a fase principal a função custo de uma aresta g é redefinida conforme abaixo:

$$custo_g = b_g + h_g \times p_g + vc_g + t_g$$

Tabela 3.9: Resultados de *overflow* e comprimento de fio do NTHU-Route para todos os circuitos do ISPD 2008.

Benchmark	NTHU-Route			NTHU-Route 2.0		
	Overflow total	WL (e5)	CPU (min)	*Overflow* total	WL (e5)	CPU (min)
adaptec1	0	54,2	79,9	0	53,4	5,2
adaptec2	0	52,8	12,6	0	52,3	1,2
adaptec3	0	132,9	47,9	0	103,5	5,3
adaptec4	0	122,7	6,9	0	121,7	1,4
adaptec5	0	157,3	215,9	0	155,4	12,2
bigblue1	0	56,2	120,7	0	56,0	7,2
bigblue2	198	90,2	48,6	0	90,6	5,7
bigblue3	18	131,6	58,1	0	130,7	3,3
bigblue4	-	-	-	150	230,9	60,4
newblue1	352	46,2	35,5	0	46,5	3,7
newblue2	0	76,3	3,0	0	75,7	0,7
newblue3	31800	107,0	303,1	31390	107,0	72,1
newblue4	448	129,1	108,7	138	130,5	60,1
newblue5	-	-	-	0	230,0	10,4
newblue6	0	178,0	214,4	0	176,9	10,4
newblue7	-	-	-	54	355,1	102,4

onde

$$t_g = \begin{cases} T_C \times (1 - e^{\beta e^{-\gamma i}}), & \text{se passar por } g \text{ entra num } \textit{hot spot} \\ 0, & \text{caso contrário} \end{cases}$$

sendo

$$T_C = \alpha \times (L_h + L_v)$$

onde L_h e L_v são as maiores cadeias horizontal e vertical de *hot spots* e α é um número entre 0 e 1 que controla a importância entre o comprimento de fio e a passagem por *hot spots*. γ e β são parâmetros definidos pelo usuário. A função custo da fase de refinamento também é alterada para a equação abaixo:

$$custo_g = \begin{cases} 1, & \text{se } g \text{ apresenta } \textit{overflow} \\ \frac{1}{TOF \times 10^n}, & \text{se passar por } g \text{ entra num } \textit{hot spot} \\ 0, & \text{caso contrário} \end{cases}$$

onde TOF é o *overflow* total após a fase principal e n é um parâmetro definido pelo usuário.

A terceira modificação é no uso de memória do roteador. O número de entradas das tabelas *hash* é melhor controlado ao longo da execução do roteador e também a limpeza de vetores usados é feita de forma mais adequada, resultando numa diminuição de 45% do uso de memória do roteador.

O roteador consegue desviar em média 14% das redes que passam pelos *hot spots* com um aumento de 4% em média no comprimento de fio e com tempo de execução 4,05x o da versão 2.0.

3.2.2 NTUgr

O roteador global NTUgr (CHEN; HSU; CHANG, 2009), segundo colocado no ISPD *Routing Contest* 2008, pode ser dividido três passos principais: pré-roteamento, rotea-

mento inicial e *rip-up and reroute* iterativo modificado. Como vários outros roteadores já citados, o NTUgr não faz roteamento 3D completo, fazendo também o mapeamento das capacidades para o plano 2D, seguido de uma fase final de assinalamento de camadas. O fluxo de execução do NTUgr é mostrado na Figura 3.29.

A etapa de pré-roteamento identifica as regiões de congestionamento para guiar as etapas seguintes. A decomposição das redes é feita usando um algoritmo de geração de MSTs, e então são aplicadas duas técnicas: pré-incremento do custo histórico nas regiões congestionadas e roteamento de redes de perímetro pequeno. O pré-incremento do custo histórico nas regiões congestionadas serve para aumentar o custo das regiões onde há grande densidade de pinos, e que, portanto, terão grande demanda de recursos de roteamento. O custo histórico é incrementado numa quantidade η, que por padrão é igual a 10, numa região de 10x10 GRCs ao redor da GRC com alta densidade de pinos. A técnica de roteamento de redes de perímetro pequeno é feita seguindo um valor limite α, que por padrão é igual a 40. Essa técnica ajuda no roteamento inicial posterior, já que redes menores possuem menos flexibilidade para roteamento, e na identificação futura das áreas com *overflow*. Nesta fase é usada a mesma função custo descrita a seguir para o roteamento monotônico.

O roteamento inicial monotônico iterativo é a primeira etapa que completa o roteamento de todas as redes do circuito. Para aumentar a eficiência do método, caminhos monotônicos são encontrados até que a redução do *overflow* da iteração $i+1$ seja menor que 5% em relação a iteração i. A função custo de um nodo x entre a origem s e o destino t é dada abaixo:

$$f(x) = g(x) + h(x) + \sum_{e \in caminho(s,x)} (1 + h_e) \times p_e$$

onde $g(x)$ é a distância Manhattan de s até x e $h(x)$ a distância Manhattan de x até t, $e \in caminho(s,x)$ representa uma aresta global no caminho de s até x e h_e e p_e são os mesmos custos histórico e de penalização já definidos anteriormente. O custo h_e só é atualizado após a última iteração, para evitar que o roteamento monotônico confunda os passos seguintes (já que este não espalha efetivamente o congestionamento por não procurar por caminhos que desviem do destino) e para evitar que o uso de A* futuro tenha seu espaço aumentado (isso porque quanto maior for o custo, maior será o espaço de busca do A*, tornando-o mais lento).

A fase seguinte é o *rip-up and rerouting* iterativo com região proibida, usado para reduzir o *overflow*, sendo a etapa mais importante do roteador. A cada iteração de *rip-up and reroute* são construídas e expandidas regiões retilíneas (regiões proibidas) a partir de regiões congestionadas, sendo aplicada uma métrica diferente de custo para estas regiões. A função custo de um nodo x entre a origem s e o destino t é dada abaixo:

$$f(x) = g(x) + h(x) + \sum_{e \in caminho(s,x)} custo_{R_e}(e)$$

onde

$$custo_{R_e}(e) = \begin{cases} custo_p(e), & \text{se } R_e \text{ está em uma região congestionada e} \\ & e \text{ está congestionada} \\ custo_h(e), & \text{caso contrário} \end{cases}$$

sendo

$$custo_p(e) = P_n \times (d_e / c_e)$$

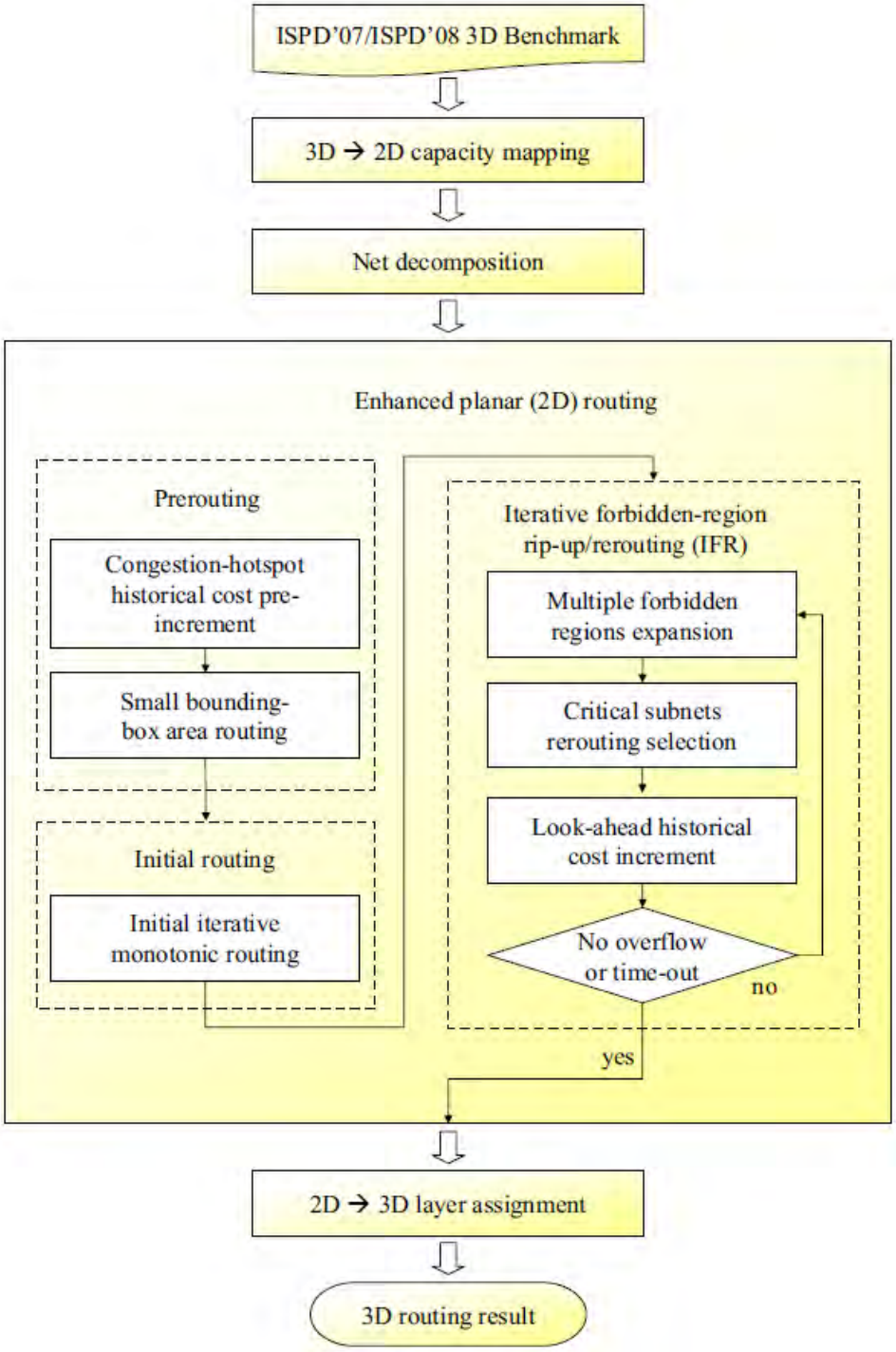

Figura 3.29: Fluxo de execução do NTUgr. (CHEN; HSU; CHANG, 2009).

onde P_n é uma penalização de *overflow* constante igual a 1000 na implementação, e

$$custo_h(e) = (1 + h_e) \times p_e$$

Os autores observaram que o *overflow* entre as iterações diminuiria mais rápido caso a diferença entre o custo das arestas sem *overflow* e as com *overflow* fosse maior. Para isso o custo p_e (normalmente igual a demanda dividida pela capacidade - d_e/c_e) foi modificado conforme a equação abaixo:

$$p_e = \begin{cases} 1/(c_e - d_e), & \text{se } c_e > d_e \\ \zeta + d_e/c_e, & \text{se } c_e < d_e \\ \zeta, & \text{se } c_e = d_e \end{cases}$$

O termo ζ tem valor padrão 3, na implementação utilizada.

O conceito de região proibida significa que introduzir *overflow* nessas regiões é quase proibido, causando um custo muito alto. São três fases para a construção destas regiões. A primeira é aplicada no início do *rip-up and reroute* e as regiões proibidas são construídas a partir da aresta com o maior *overflow* e que ainda não estejam em nenhuma região proibida. Inicialmente a região proibida contem apenas duas GRCs adjacentes (que compartilham a aresta mais congestionada), sendo expandida a fronteira mais congestionada da região, até que o critério de parada seja atingido. Para uma região proibida R_f, o congestionamento médio nas quatro fronteiras de R_f como:

$$C^d_{R_f} = \frac{\sum d_e}{\sum c_e}, \forall e \text{ na fronteira } d \text{ de } R_f$$

onde $d \in \{L, R, T, B\}$ representam as fronteiras esquerda, direita, superior e inferior de R_f, e e representa a aresta global situada nas fronteiras de R_f. O critério de parada é máx$\{C^L_{R_f}, C^R_{R_f}, C^T_{R_f}, C^B_{R_f}\} < \beta$, onde β é um limite que diminui com o aumento das iterações de *rip-up and reroute*, variando entre 1,4 e 1,1. Como a expansão das regiões proibidas é múltipla, estas podem se sobrepor formando polígonos retilíneos, como mostrado na Figura 3.30.

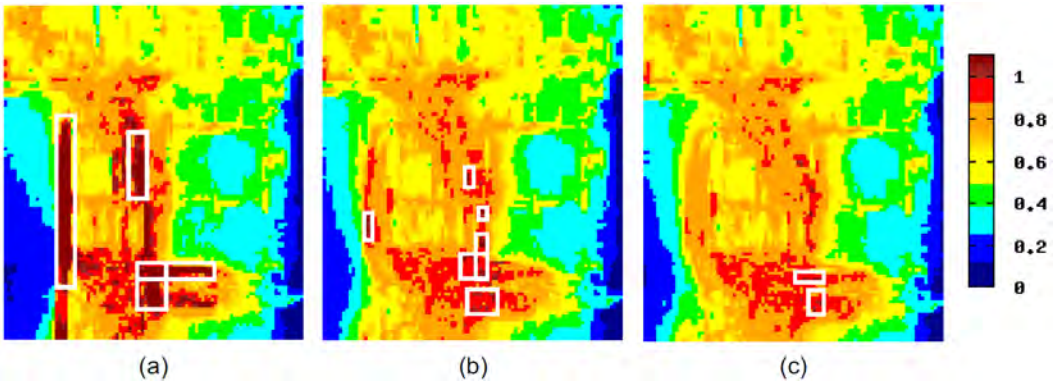

Figura 3.30: Regiões proibidas ao longo das iterações (a) i, (b) $i + 1$ e (c) $i + 2$, para o circuito adaptec5. (CHEN; HSU; CHANG, 2009).

A segunda fase é iniciada quando o número de *overflows* na primeira fase para de diminuir e fica preso num mínimo local da solução. Nesta fase foi desenvolvida uma nova técnica chamada nivelamento da propagação de região (*region propagation levelling*

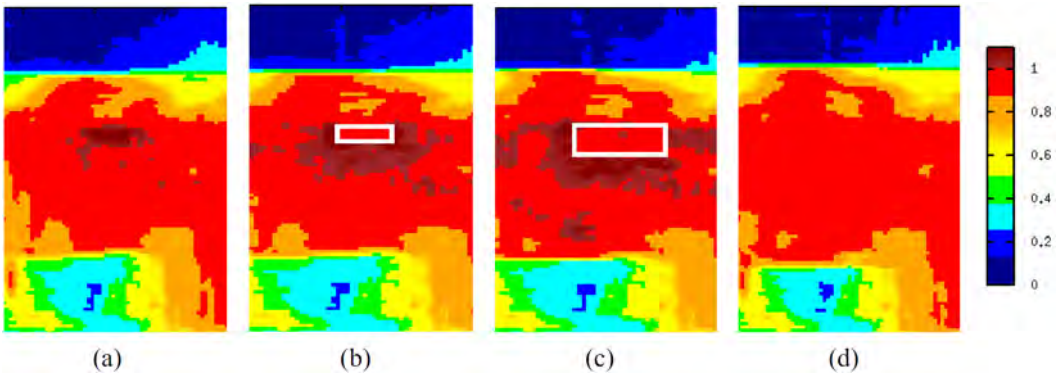

Figura 3.31: Congestionamento no circuito bigblue3 na fase de RPL, para a iteração (a) i, (b) $i+1$, (c) $i+2$ e (d) ao final das iterações (sem *overflow*). (CHEN; HSU; CHANG, 2009).

- RPL) que, diferente da fase anterior onde as regiões proibidas eram reconstruídas a cada iteração, o RPL herda as regiões proibidas das iterações anteriores e expande-as simultaneamente. Isso serve para evitar que o congestionamento fique ao redor das regiões proibidas, como pode ser visto na Figura 3.31.

A terceira fase inicia quando o número de *overflows* é menor que 0,5% do total de *overflows* após a fase de roteamento inicial monotônico. Nessa fase, o *rip-up and re-route* iterativo aplica outra técnica chamada expansão final (*final expasion*), que expande a região proibida até cobrir todo o circuito para reduzir rapidamente os *overflows* remanescentes. Essa técnica é usada para que o número de *overflows* convirja para zero mais rapidamente, como mostrado na Figura 3.32.

Outra técnica adicionada ao *rip-up and rerouting* iterativo com região proibida é chamada de seleção de sub-redes críticas para re-roteamento(*critical subnets rerouting selection*), que visa reduzir o número de redes a serem re-roteadas a cada iteração, já que o *rip-up and rerouting* é a etapa mais demorada da execução da ferramenta, sendo apenas desfeitas e re-roteadas as redes (n) que são consideradas críticas, segundo a equação abaixo:

$$\min_{e \in n} \{c_e - d_e\} \leq S$$

onde S é uma constante com valor padrão -1 e e é a aresta passada por n. Este valor padrão de S significa que apenas as redes com *overflow* serão re-roteadas, o que representou um desempenho 1,21x mais rápido que sem o uso dessa técnica.

A última técnica adicionada é o o incremento adiantado do custo histórico, que nada mais é do que fazer o incremento do custo histórico não só das arestas com *overflow* mas também de arestas que estejam próximas de ter *overflow*, aplicando a função de custo histórico abaixo:

$$h_e^{i+1} = \begin{cases} h_e^i + K, & \text{se } d_e + N > c_e \\ h_e^i, & \text{caso contrário} \end{cases}$$

onde $N > 0$ é um inteiro com valor igual a 1 (mesmo valor de K), que reflete numa melhora substancial na qualidade do roteador.

A etapa final do roteador é a fase de assinalamento de camadas 2D para 3D, que é implementado com uma técnica programação dinâmica, onde o primeiro critério de custo é o número de *overflows* e o segundo critério é o número de vias e fios utilizados, sendo

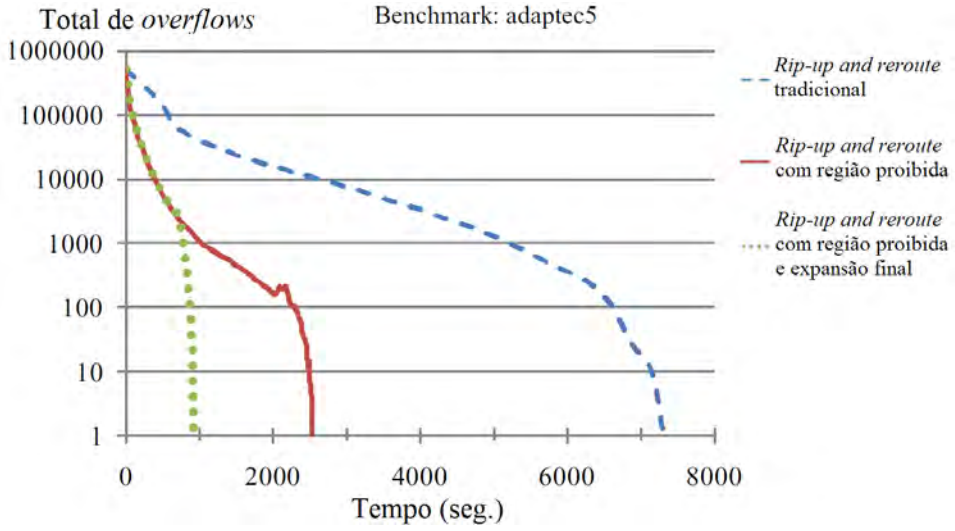

Figura 3.32: Número de *overflows* versus tempo de execução com e sem as técnicas descritas, para o circuito adaptec5. Adaptada de (CHEN; HSU; CHANG, 2009).

também preferido o compartilhamento de arestas e vias pelas sub-redes através do custo zero para o compartilhamento.

Os resultados do NTUgr para os circuitos do ISPD 2008 são mostrados na Tabela 3.10.

Tabela 3.10: Resultados de *overflow* e comprimento de fio do NTUgr para todos os circuitos do ISPD 2008.

Benchmark	*Overflow* total	*Overflow* máx.	WL (e5)	CPU (min)
adaptec1	0	0	57,4	4,5
adaptec2	0	0	53,7	1,1
adaptec3	0	0	135,0	4,4
adaptec4	0	0	123,7	1,2
adaptec5	0	0	159,9	15,3
bigblue1	0	0	60,0	18,1
bigblue2	0	0	91,2	248,3
bigblue3	0	0	133,5	4,0
bigblue4	188	8	242,8	413,1
newblue1	6	2	49,3	977,5
newblue2	0	0	76,9	0,6
newblue3	31024	408	188,3	884,0
newblue4	142	2	143,8	1118,1
newblue5	0	0	244,9	20,5
newblue6	0	0	186,6	21,3
newblue7	310	2	372,2	1445,5

3.2.3 FastRoute

O FastRoute (PAN; CHU, 2006, 2007; ZHANG; XU; CHU, 2008; XU; ZHANG; CHU, 2009), cuja versão 3.0 foi a terceira colocada no ISPD *Global Routing Contest* 2008, foi primeiramente proposto como um estimador de congestionamento para a fase de posicionamento, apresentando resultados melhores que os roteadores globais até então, o

Labyrinth (KASTNER; BOZORGZADEH; SARRAFZADEH, 2000) e o *Chi Dispersion* (HADSELL; MADDEN, 2003). Foi o primeiro trabalho a propor e utilizar a técnica de *edge shifting*, já discutida anteriormente.

A primeira versão da ferramenta apresenta três fases principais: geração de mapa de congestionamento, construção de árvores de Steiner dirigidas a congestionamento e roteamento das redes de dois pinos usando *maze routing* e *pattern routing*.

Na primeira fase as árvores de Steiner são construídas usando o FLUTE, sendo estas então quebradas em redes de dois terminais e roteadas utilizando roteamento em forma L (*L-shaped pattern routing*), resultando no mapa de congestionamento inicial.

A construção de árvores de Steiner dirigidas a congestionamento é feita através do uso de duas técnicas propostas. Primeiro, um algoritmo de geração de topologias dirigidas a desempenho gera as árvores de Steiner para reduzir o congestionamento de acordo com o mapa de congestionamento anterior. Isso é feito tornando a distância entre linhas da grade de Hanan (HANAN, 1966) proporcionais ao congestionamento médio da região, como exemplificado na Figura 3.33, onde os valores de $v2$ e $h2$ são escalados de acordo com a média de congestionamento da região. Após o escalamento das regiões, o FLUTE é utilizado novamente para encontrar as RSMTs em termos dessas novas distâncias calculadas, resultando em melhor distribuição do congestionamento. A segunda técnica é a de deslocamento de arestas (*edge shifting*), onde arestas relacionadas a nodos de Steiner podem ser deslocadas sem que haja mudança no comprimento de fio da rede. Na Figura 3.34 é exemplificado como é feita a definição do alcance máximo de deslocamento que a aresta pode sofrer, sendo que somente arestas cujos terminais tenham grau 3, ou seja, ligados a três arestas, podem ser deslocadas. Como nodos de Steiner podem ser de grau 3 ou 4, os terminais de grau 4 são quebrados em dois de grau 3. As arestas são deslocadas sempre para a posição que resultar em menor congestionamento, sendo esta técnica aplicada para todas as redes mais de quatro terminais.

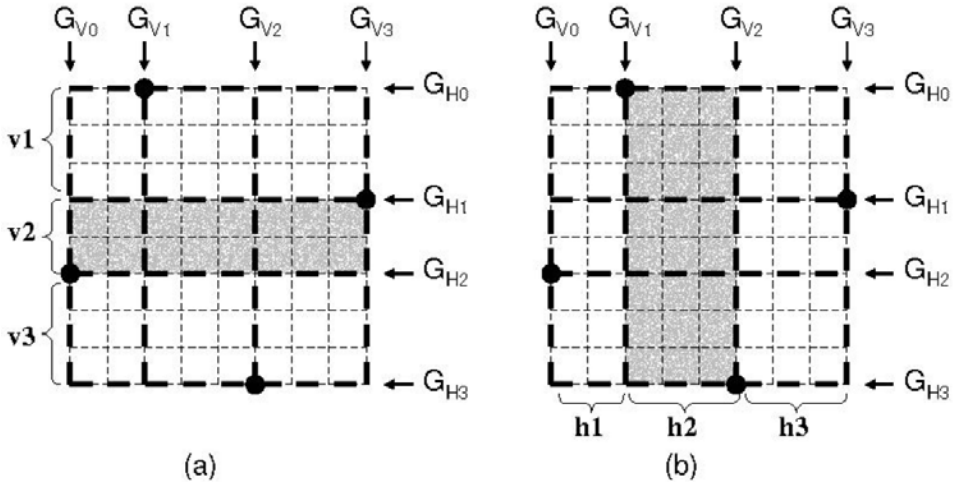

Figura 3.33: Exemplo de delimitação das regiões a serem escaladas, na vertical (a) e na horizontal (b). (PAN; CHU, 2006).

A fase final é o roteamento de todas as redes de dois pinos obtidas na fase anterior com *pattern routing* e *maze routing*. Primeiramente, o *pattern routing* é feito com conexões em forma de Z, sendo então depois aplicado *rip-up and reroute* usando *maze routing*, como

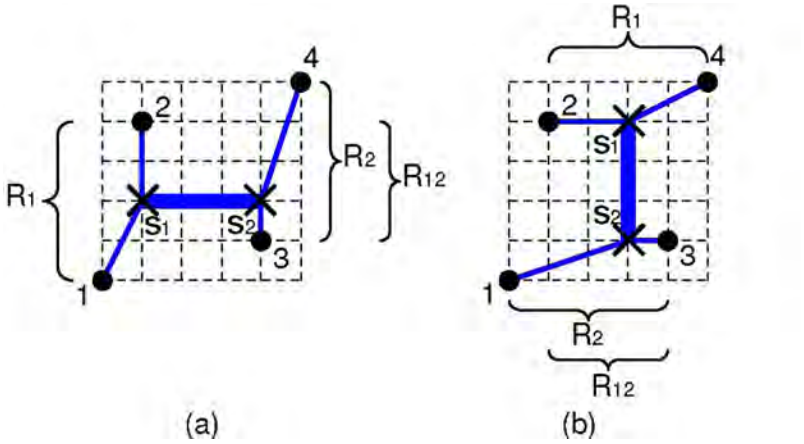

Figura 3.34: Definição do alcance de deslocamento da aresta. (PAN; CHU, 2006).

visto em outros trabalhos. A função custo utilizada nesta etapa é a mostrada abaixo:

$$custo = 1 + \frac{h}{1 + e^{-k(demanda - capacidade)}}$$

onde h e k são parâmetros da função que controlam a magnitude do custo e a inclinação da curva, respectivamente, sendo que ambos não têm seus valores especificados pelos autores.

No FastRoute 2.0 duas alterações são propostas na terceira fase descrita acima para obter resultados com melhor qualidade: o uso de roteamento monotônico em vez de *pattern routing* e o uso de *maze routing* com múltiplas origens e múltiplos destinos (*multi-source multi-sink maze routing* - MMMR).

O roteamento monotônico é feito usando programação dinâmica e, apesar de possuir a mesma complexidade que o *pattern routing* com formas Z ($O(mn)$), mostrou-se na prática 2,3x mais lento que o anterior.

O *maze routing* com múltiplas origens e múltiplos destinos trata como origens todos os pontos da grade da subárvore onde está o nodo de origem e trata como destinos todos os pontos da grade da subárvore de destino, evitando que caminhos não ótimos sejam gerados usando apenas os dois pinos como origem e destino. O MMMR é usado apenas para redes maiores que um certo limite definido (mas não especificado no trabalho).

Já o FastRoute 3.0 introduz o conceito de capacidade virtual (*virtual capacity*), onde a capacidade de roteamento de arestas com *overflow* é reduzida para que o roteamento por esta fique mais caro.

A técnica de capacidade virtual consiste de duas ideias: inicialização da capacidade virtual e atualização da mesma. A inicialização da capacidade virtual é feita usando uma estimativa de congestionamento adaptativa (*adaptative congestion estimation* - ACE), que considera o roteamento apenas dentro do perímetro da rede (*bounding box*). Após essa estimativa, a capacidade virtual vc de cada aresta é calculada pela equação abaixo:

$$vc_e = rc_e - (max(0, p_e - rc_e))\ \forall e$$

onde rc é a capacidade real da aresta e p_e é a estimativa de uso calculada pelo ACE. A atualização das capacidades virtuais é necessária pois após alguma iterações ela fica menos efetiva, pois o *maze routing* cria desvios nas redes não previstos pelo ACE. Assim

a capacidade virtual das arestas é atualizada conforme a equação abaixo:

$$vc_e = \begin{cases} vc_e - o_e, & \text{se } o_e \geq 0 \\ vc_e - F \times o_e, & \text{se } o_e < 0 \end{cases}$$

onde

$$o_e = u_e - c_e$$

sendo u_e a demanda na aresta e. o Parâmetro F é por padrão 0,85 e serve para aumentar a capacidade virtual após essa ter diminuído e não ter mais *overflow*.

Nesta versão também é incorporado o custo das vias durante o *maze routing*, sendo que o custo unitário da via é adicionado sempre que é gerado um desvio, e uma função custo adaptativa para *maze routing*, mostrada abaixo:

$$custo_e = \begin{cases} 1 + \frac{H}{1+e^{-k(demanda-capacidade)}}, & \text{se } 0 < u_e \leq c_e \\ 1 + M + S \times (u_e - c_e), & \text{se } u_e > c_e \end{cases}$$

onde H é a "altura"do custo (que determina sua magnitude, sendo aumentado a cada iteração), k controla a inclinação da curva de custo (quanto maior mais vai provocar desvios, ou seja, maior comprimento de fio), M é o custo quando a demanda é igual à capacidade e S determina a inclinação da curva quando há *overflow*.

Por fim há a etapa de assinalamento de camadas para gerar as soluções para circuitos 3D.

Na versão 4.0 são agregadas três técnicas para redução do uso de vias: geração de árvore de Steiner ciente de vias, roteamento com três desvios e assinalamento de camadas com ordenamento cuidadoso de redes e arestas. Os estágios hachurados na Figura 3.35 mostram as modificações em relação ao fluxo do FastRoute 3.0.

A geração de árvores de Steiner ciente de vias produz topologias de árvores que ajusta o número de vias baseado nas informações de congestionamento e camadas. Isso é feito utilizando as capacidades e uso das camadas de metal conforme a equação abaixo:

$$\text{fator de via} = \frac{\sum cap_{(h)}}{\sum uso_{(h)}} / \frac{\sum cap_{(v)}}{\sum uso_{(v)}}$$

Esse fator é multiplicado ao fator de escala de congestionamento vertical usado na geração das redes pelo FLUTE, descrito anteriormente, assim, por exemplo, se houver mais recursos horizontais disponíveis as distâncias verticais escaladas serão maiores o que resultará em mais fios horizontais. Isso porque árvores predominantemente horizontais ou verticais representam os dois extremos de número de vias para uma determinada sequência de camadas. A Figura 3.36 mostra uma árvore preferencialmente horizontal (H-tree), onde há apenas um segmento vertical e todos outros são horizontais, e uma árvore preferencialmente vertical (V-tree), onde há apenas um segmento horizontal e todos outros verticais e uma SMT. Como a camada de metal 1 neste exemplo é horizontal e considerando que o roteamento se dará nas duas camadas adjacentes (metal 1 e metal 2), a H-tree apresenta o valor mínimo de vias (sete), a V-tree o valor máximo (14) e a SMT um valor intermediário (nove). Como os recursos de metal 1 podem não estar disponíveis, a H-tree deixaria de ser a melhor opção, já que o roteamento teria que ser feito nas camadas de metal 2 e metal 3, fazendo com que a V-tree fosse a opção com menos vias, mostrando que a relação das capacidades disponíveis influi no número de vias. Essa técnica apresentou melhorias de 3% no número total de vias com impacto de 1% no comprimento de fio e no *overflow*.

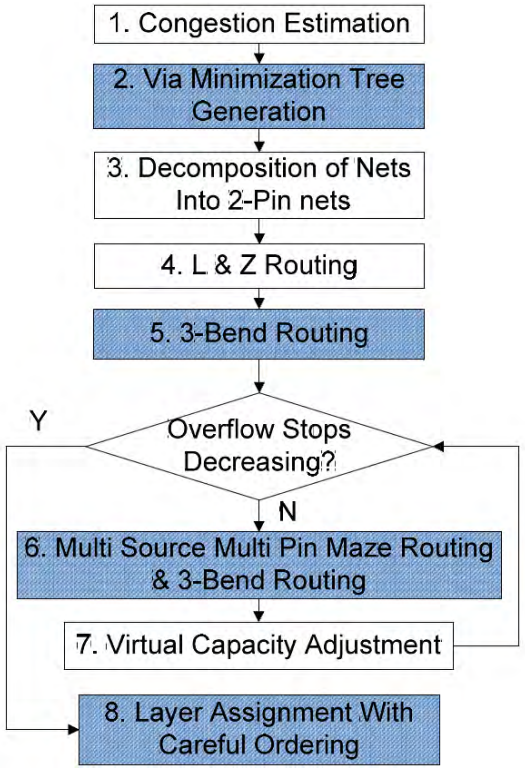

Figura 3.35: Fluxo do FastRoute 4.0. (XU; ZHANG; CHU, 2009).

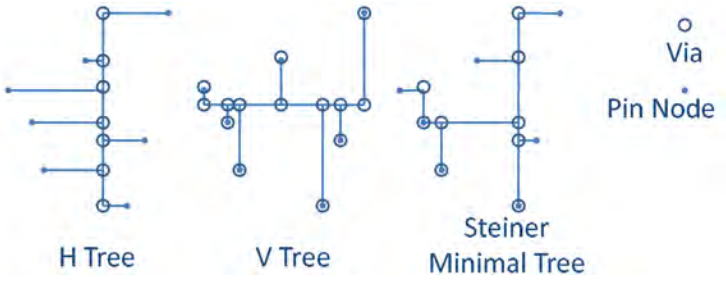

Figura 3.36: Topologias diferentes com diferentes número de vias. (XU; ZHANG; CHU, 2009).

O roteamento com três desvios (*3-bend routing*) é um algoritmo de roteamento eficiente com opção de fazer desvios em regiões congestionadas. Esse tipo de roteamento permite maior flexibilidade para as redes desviarem de regiões de maior congestionamento sem grande tempo de execução como *maze routing*. Esse roteamento é feito em uma região expandida do perímetro da rede de acordo com o tamanho, localização e congestionamento de cada rede. Essa técnica se torna mais efetiva com o aumento da complexidade dos circuitos, evitando o aumento excessivo do uso de *maze routing*.

Para o assinalamento de camadas é usado um algoritmo sequencial que cuidadosamente ordena redes e as arestas de cada rede, usando programação dinâmica. Redes menores são consideradas mais locais, devendo, portanto, ser assinaladas às camadas mais inferiores, enquanto redes maiores são mais adequadas às camadas superiores para evitar muitos "pulos"entre camadas, já que as inferiores são mais concorridas. Além disso, notou-se que redes com mais pinos tendem a ter maior número de vias, por isso, as redes são ordenadas crescentemente pelo somatório do comprimento de fio dividido pelo

número de pinos da rede, fazendo com que as redes menores e com mais pinos sejam roteadas nas camadas mais inferiores de metal. Cada rede tem suas arestas ordenadas em ordem crescente de distância dos pinos, ou seja, o assinalamento é feito a partir dos pinos até as regiões centrais da rede.

Os resultados das três primeiras versões do FastRoute para os *benchmarks* do ISPD 1998 são mostrados na Tabela 3.11 e os resultados para os circuitos do ISPD 2008 são mostrados na Tabela 3.12.

Tabela 3.11: Resultados de *overflow* e comprimento de fio do FastRoute para todos os circuitos do ISPD 1998.

Benchmark	FastRoute			FastRoute 2.0			FastRoute 3.0		
	Ovfl	WL	CPU (s)	*Ovfl*	WL	CPU (s)	*Ovfl*	WL	CPU (s)
ibm01	250	67128	0,21	31	68489	0,72	0	64221	0,64
ibm02	39	179995	0,56	0	178868	0,93	0	172223	0,85
ibm03	1	151023	0,43	0	150393	0,60	0	146753	0,49
ibm04	567	172593	0,50	64	175037	1,88	0	170146	2,70
ibm06	33	285882	0,91	0	284935	1,36	0	279471	1,15
ibm07	18	376835	1,05	0	375185	1,60	0	369023	1,68
ibm08	58	412915	1,16	0	411703	2,36	0	405935	1,82
ibm09	28	426471	1,39	3	424949	1,92	0	414913	1,67
ibm10	18	599433	1,98	0	595622	2,79	0	582838	3,61

Tabela 3.12: Resultados de *overflow* e comprimento de fio do FastRoute para todos os circuitos do ISPD 2008. Tempo de CPU em minutos. *valores medidos pela métrica do ISPD 2007. **versão do concurso.

Benchmark	FastRoute 3.0			FastRoute 3.0c**			FastRoute 4.0		
	Ovfl	WL (e5)	CPU	*Ovfl*	WL (e5)	CPU	*Ovfl*	WL (e5)	CPU
adaptec1	0	55,2	5	0	55,5	2	0	91,00*	3,17
adaptec2	0	53,7	1	0	53,1	1	0	91,57*	0,75
adaptec3	0	133	4	0	133	2	0	204,61*	5
adaptec4	0	123	1	0	122	1	0	186,95*	0,67
adaptec5	0	161	11	0	161	5	0	270,64*	9,47
bigblue1	0	59,5	7	0	58,3	4	0	57,89	7,05
bigblue2	0	98,8	16	142	98,2	11	0	95,59	15,22
bigblue3	0	132	2	0	132	3	0	130,70	4,63
bigblue4	156	244	35	206	243	41	152	241,65	11,23
newblue1	0	48,2	5	76	49	13	0	91,70*	4,72
newblue2	0	76,3	1	0	76,2	1	0	135,63*	0,37
newblue3	31634	110	36	31650	109	31	31634	182,11*	21,33
newblue4	154	137	17	226	136	10	144	133,94	18,92
newblue5	0	240	11	0	241	5	0	236,72	10,12
newblue6	0	186	10	0	187	4	0	182,62	9,57
newblue7	108	361	160	588	359	190	62	356,91	184,33

3.3 Trabalhos Recentes

Após a realização dos dois concursos de roteamento global muitos outros trabalhos foram publicados, abordando não só o problema de congestionamento mas também outros focos, temperatura do circuito e número de vias (problema já tratado pelo roteadores dos concursos devido ao seu custo nas métricas de classificação).

Apesar deste trabalho ter foco nos roteadores propostos nos concursos sobre roteamento global, outros trabalhos importantes foram publicados mesmo antes da realização destes concursos, além das novas versões dos roteadores globais participantes dos concursos já apresentadas nas seções 3.1 e 3.2. Com o intuito de delinear corretamente o estado da arte, esses trabalhos serão discutidos a seguir, com enfoque maior para os que possuem os melhores resultados.

3.3.1 GRIP

O GRIP (WU; DAVOODI; LINDEROTH, 2009, 2010, 2011) é um roteador global baseado em ILP que propõe uma nova formulação para o problema de roteamento, diferente da formulação ILP já vista no BoxRouter. Essa formulação permite que haja um roteamento 3D completo, através da consideração do custo das vias e comprimento de fio em 3D, evitando a fase de assinalamento de camadas. A formulação é mostrada a seguir. Dado o grafo $G = (V, E)$, sendo $\mathcal{N} = T_1, T_2, ..., T_N$ a lista de redes, u_e a capacidade das arestas, c_e o custo das arestas, $\mathcal{T}(T_i)$ o conjunto das topologias de rede que conectam T_i e o parâmetro $a_{te} = 1$ se a árvore t contem a aresta e e $a_{te} = 0$ caso contrário, e x_{it} a variável de decisão binária que será igual a 1 se e somente se a rede T_i for roteada com o caminho $t \in \mathcal{T}(T_i)$.

$$\min_{x,s} \sum_{i=1}^{N} \sum_{t \in \mathcal{T}(T_i)} c_{it} x_{it} + \sum_{i=1}^{N} M s_i \qquad \text{ILP-GR}$$

$$\sum_{t \in \mathcal{T}(T_i)} x_{it} + s_i = 1 \quad \forall i = 1, \ldots, N \qquad (1)$$

$$\sum_{i=1}^{N} \sum_{t \in \mathcal{T}(T_i)} a_{te} x_{it} \leq u_e \quad \forall e \in E \qquad (2)$$

$$x_{it} \in \{0, 1\} \quad \forall i = 1, \ldots, N \quad \forall t \in \mathcal{T}(T_i)$$

$$s_i \geq 0 \quad \forall i = 1, \ldots, N.$$

onde c_{it} é o custo do caminho t para a rede T_i (custo do caminho 3D), s_i é a variável de decisão que será positiva se a rede T_i não puder ser roteada, sendo M o seu peso, usado com valor suficientemente grande para evitar que redes fiquem não roteadas (valor 20000 na implementação).

Neste caso a ILP também é usada para a geração de possíveis caminhos diferentes, pelo método de geração de colunas, onde são geradas soluções com menor custo que a rede candidata atual e armazenadas para posterior avaliação também usando formulação ILP.

Em roteamento, esse problema corresponde ao problema de custo (*pricing problem*), onde uma solução (rede) de menor custo deve ser achada a cada iteração da geração de colunas, se esta existir. Durante esta fase é usado *maze routing* para cada segmento de

dois terminais para encontrar caminhos de menor custo. As árvores iniciais (primeiro roteamento candidato) são geradas usando o FLUTE.

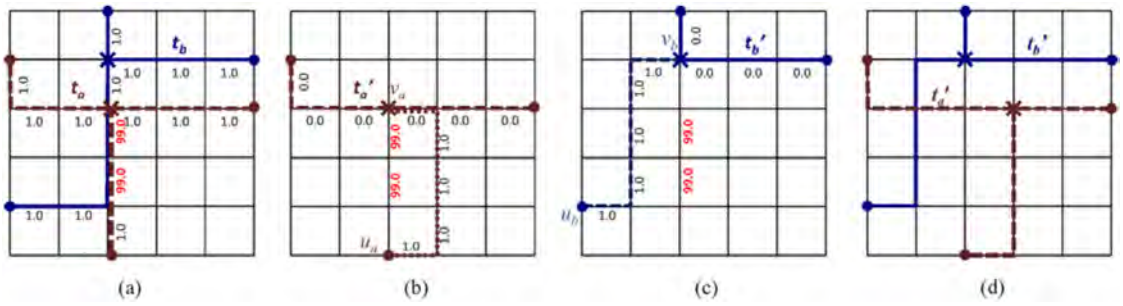

Figura 3.37: Geração das redes candidatas. (WU; DAVOODI; LINDEROTH, 2011).

Um exemplo de como isso é feito é mostrado na Figura 3.37 (a), onde as redes T_a e T_b têm roteamento inicial t_a e t_b, respectivamente, onde a capacidade das arestas é unitária (custo também unitário) e a penalização de *overflow* é de 99, portanto, ambas as redes têm custo total 205. Assumindo que a rede T_a é escolhida primeiro para a avaliação de custo, a subárvore que possui as duas arestas com *overflow* é desfeita primeiro e re-roteada conforme a Figura 3.37 (b), passando a ter custo dez[2]. O mesmo procedimento é executado para a rede T_b, conforme a Figura 3.37 (c). Assim, as duas novas árvores encontradas, que possuem custo menor que a inicial, são adicionadas à lista de candidatas para a resolução do problema ILP de roteamento.

Como o número de redes é grande, apenas algumas redes são escolhidas para avaliação de custo. Estas são escolhidas de acordo com o congestionamento que apresentam, usando as informações geradas pelo próprio cálculo do dual do problema mestre restrito (*restricted master problem*) da formulação ILP e selecionando um perímetro de 3×3 arestas da grade em torno das arestas selecionadas para serem avaliadas. Esse processo é repetido até que nenhum caminho de menor custo é encontrado para as redes selecionadas ou até que haja estagnação na melhoria da solução (se a função objetivo da formulação tem nenhuma ou pouca melhoria, no caso, 10 unidades de comprimento de fio, nas últimas 20 iterações). Essas redes candidatas encontradas são então passadas para o resolvedor ILP[3]. Esse fluxo de execução é mostrado na Figura 3.38.

Como o problema de roteamento costuma ser muito grande, a resolução do problema ILP não pode ser aplicada diretamente em todo o circuito. Para contornar esse problema, o circuito é dividido em sub-regiões, sendo as redes que atravessam as fronteiras destas regiões representadas por vértices e arestas auxiliares (pseudo-terminal), como mostrado na Figura 3.39, onde as arestas em negrito representam as fronteiras e os triângulos os nodos flutuantes da rede que atravessa a fronteira. As arestas adicionadas têm capacidade infinita e custo zero na formulação ILP.

A divisão das sub-regiões é feita a partir do mapa utilização inicial das arestas de acordo com as redes geradas pelo FLUTE. Um algoritmo de bi-particionamento é aplicado de modo a manter sempre a mesma utilização de aresta média em cada uma das duas regiões criadas. Para escolher entre um corte vertical ou horizontal o GRIP escolhe o que

[2]similar a outros roteadores, o algoritmo atribui custo zero para as arestas que também fazem parte das outras subárvores.

[3]pacote CPLEX para programação inteira e pacote MOSEK para programação linear, ambos da linguagem C++.

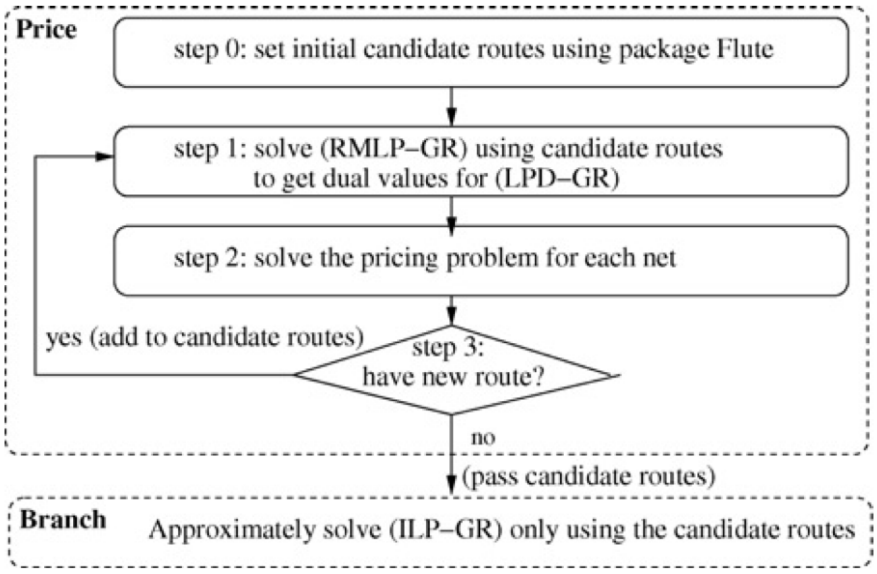

Figura 3.38: Fluxo de execução do GRIP. (WU; DAVOODI; LINDEROTH, 2011).

apresenta a menor relação de aspecto dos retângulos gerados. O algoritmo para quando um dos lados da partição atual alcança 32 unidades da grade de roteamento.

Antes de resolver cada subproblema, o GRIP desvia o máximo possível de redes das sub-regiões mais congestionadas (somente as que não possuem pinos dentro da sub-região congestionada), como mostrado na Figura 3.40. Isso é feito usando *maze routing* com o peso dos custos das arestas da sub-região A com valor infinito, das arestas sem capacidade disponível com valor igual a 100 e as demais com peso unitário.

Para reconectar as sub-regiões, o GRIP desfaz o caminho que passa pela fronteira (caminho que conecta ao pseudo-terminal) e refaz essa rota considerando as duas sub-regiões sendo conectadas, usando a mesma formulação ILP usada anteriormente.

Após conectar todas as sub-regiões, GRIP avalia se todas as redes então conectadas, e se este for o caso, ele é encerrado. Caso contrário, uma etapa de redução de *overflow* é aplicada. Para esta etapa também é feita uma formulação ILP, só que desta vez com a função objetivo de simplesmente reduzir o *overflow*, desconsiderando o comprimento de fio. O procedimento de definição das sub-regiões nesta etapa é de acordo com o *overflow* médio, sendo as sub-regiões muito maiores, já que a maior parte do problema já foi resolvido.

Os resultados do GRIP para s *benchmarks* do ISPD 2008 são mostrados na Tabela 3.13. Em média o GRIP apresenta comprimento de fio em torno de 6% menor que o NTHU-Route 2.0 e que o FGR 1.1, gerando soluções legais para os mesmos 12 circuitos que o NTHU-Route 2.0. O GRIP explora o paralelismo de CPU para as sub-regiões, resultando num tempo médio de CPU (soma de todos os tempos dos processamentos paralelos) de 68 horas em média, e um tempo médio de execução de 10,5 horas para os circuitos testados, para uma grade de centenas de diferentes CPUs com 2GB de memória, gerenciados pelo sistema de gerenciamento Condor.

3.3.2 Archer

O Archer (OZDAL; WONG, 2007, 2009) é um roteador global baseado em *rip-up and reroute* que faz a otimização da topologia das redes de acordo com o congestionamento, utilizando relaxação lagrangiana. Para circuitos 3D, o Acher também usa a técnica de

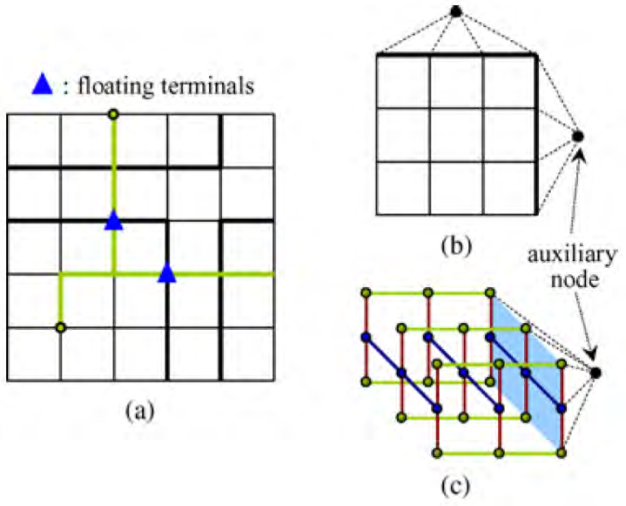

Figura 3.39: Representação de redes que atravessam a(s) fronteira(s) das sub-regiões. (WU; DAVOODI; LINDEROTH, 2011).

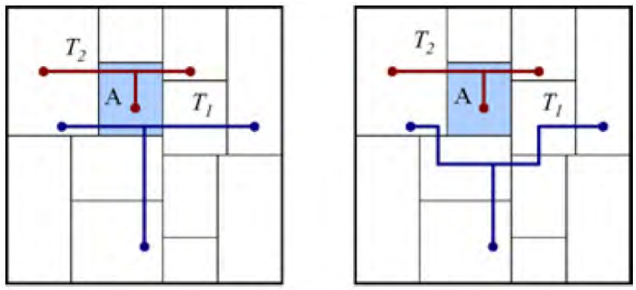

Figura 3.40: Deslocamento da rede T_1 para evitar a sub-região congestionada A. (WU; DAVOODI; LINDEROTH, 2011).

mapeamento para 2D e fase final de assinalamento de camadas.

Inicialmente, o Archer usa o FLUTE para gerar as SMTs, roteando todas as redes com o menor comprimento de fio possível como solução inicial. Após isso, são executadas iterações de *rip-up and reroute* sobre as redes decompostas em redes de dois pinos, até que uma solução sem *overflow* seja encontrada ou até um certo número máximo de iterações. Cada iteração busca re-rotear as redes minimizando a equação abaixo:

$$custo(e) = (1 + \alpha \times h_e^k) \times overflow(e)$$

onde e é a aresta atual e k o número de iterações, sendo

$$h_e^k = \begin{cases} h_e^k, & \text{se a aresta } e \text{ não possui } \textit{overflow} \\ h_e^k + k, & \text{se a aresta } e \text{ possui } \textit{overflow} \end{cases}$$

O parâmetro α é o fator de escala do custo histórico, que determina a importância deste perante o *overflow*. Durante a fase inicial o valor do histórico não é muito preciso, por isso, a minimização de *overflow* é priorizada, sendo então α um valor pequeno, que é aumentado gradualmente a cada iteração. Na fase de negociação α tem valor máximo, fazendo com que apenas as redes que realmente necessitem dos recursos com *overflow* mantenham-se nestes, já que o custo histórico aumenta a cada iteração. Na fase final de

Tabela 3.13: Resultados de *overflow* e comprimento de fio do GRIP para os circuitos do ISPD 2008. *resultados de comprimento de fio usando a métrica do ISPD 2007

Benchmark	GRIP		GRIP c/ etapa de red. *OF*	
	Ovfl	WL (e5)	*Ovfl*	WL (e5)
adaptec1	0	81,0*	0	-
adaptec2	0	82,4*	0	-
adaptec3	0	185,4*	0	-
adaptec4	0	172,3*	0	-
adaptec5	0	238,9*	0	-
bigblue1	0	53,7	0	-
bigblue2	0	86,0	0	-
bigblue3	0	126,2	0	-
bigblue4	196	220,5	180	220,7
newblue1	0	83,9*	0	-
newblue2	0	121,4*	0	-
newblue3	52518	156,1*	45960	157,6
newblue4	196	124,2	136	124,4
newblue5	0	222,8	0	-
newblue6	0	170,5	0	-
newblue7	110	335,5	54	335,8

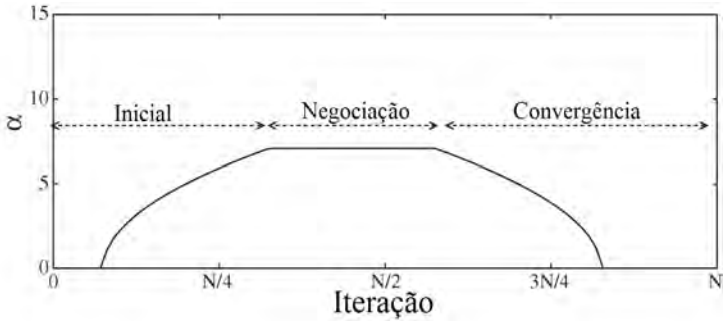

Figura 3.41: Função de α ao longo das iterações. (OZDAL; WONG, 2009).

convergência, α é novamente reduzido para que o foco volte a ser a redução do *overflow*. Os valores de α ao longo destas fases são mostrados na Figura 3.41.

O roteamento das redes durante o *rip-up and reroute* é feito utilizando, nesta ordem: *pattern routing* de comprimento de fio mínimo (formas I/L/Z), *pattern routing* com desvios (forma U), *maze routing* monotônico e *maze routing* não monotônico. Cada uma dessas técnicas é aplicada caso a anterior tenha sido infrutífera em encontrar um roteamento sem *overflow* por repetidas iterações. O objetivo é evitar o máximo possível que redes sejam roteadas com *maze routing*, já que este exige maior tempo de execução. Para evitar que redes com vários terminais tenham sobreposição das suas sub-redes de dois terminais, as arestas já ocupadas por sub-redes da mesma rede têm custo zero, como já descrito em outros roteadores.

A otimização da topologia das redes dirigida de acordo com o congestionamento é feita por uma formulação de multiplicadores de Lagrange, conforme abaixo:

$$\text{minimizar: } \sum_{e \in T} custo(e) \text{desde que: } \sum_{e \in T} comprimento(e) \leq L_{max}$$

onde T é a topologia a ser obtida e e é a aresta da grade de Hanan em T. Assim a função de Lagrange é a mostrada abaixo:

$$F(\lambda) = \sum_{e \in T} custo(e) + \sum_{e \in T} \lambda.comprimento(e)$$

onde λ é o valor do multiplicador de Lagrange correspondente à restrição de comprimento, que é atualizado a cada iteração. Intuitivamente, se o comprimento de T é maior que L_{max} após uma iteração, então λ é aumentado para priorizar a minimização do comprimento, e vice versa. Dado λ^k numa iteração k, λ^{k+1} será:

$$\lambda^{k+1} = \max(0, \lambda^k + t_k) \times \begin{cases} 1, & \text{se } comprimento(T) > L_{max} \\ -1, & \text{se } comprimento(T) \leq L_{max} \end{cases}$$

onde $t_k = 1/k^\alpha$ e $\alpha = 0,1$.

Após a solução de roteamento estar pronta, aplica-se sobre esta o assinalamento de camadas, com o objetivo de minimizar o número de vias. Primeiramente, as redes são ordenadas em ordem crescente de comprimento de fio, então, dois passos são aplicados sobre o roteamento 2D. O primeiro passo tenta assinalar cada segmento s para uma camada (de forma planar) desde que: o número de vias seja minimizado e que a capacidade das arestas não sejam ultrapassadas. Caso o segmento não possa ser assinalado ele fica na lista de segmentos não roteados. O segundo passo processa os segmentos não roteados, roteando-os (não necessariamente de forma planar) desde que: o número de vias seja minimizado, a forma do roteamento 2D original seja preservada e a capacidade das arestas 3D não seja ultrapassada.

Os resultados do Archer para os circuitos do ISPD 1998 são mostrados na Tabela 3.14 e para os circuitos do ISPD 2007 na Tabela 3.15.

Tabela 3.14: Resultados de *overflow* e comprimento de fio do Archer para todos os circuitos do ISPD 1998.

Benchmark	Archer		
	Ovfl	WL	CPU (s)
ibm01	0	64389	11
ibm02	0	171805	25
ibm03	0	146770	10
ibm04	0	169977	24
ibm05	0	409761	8
ibm06	0	278841	23
ibm07	0	370143	25
ibm08	0	404530	42
ibm09	0	414223	37
ibm10	0	583805	45

3.3.3 BFG-R

O BFG-R (HU; ROY; MARKOV, 2010), evolução do trabalho do FGR, tem como objetivo minimizar o comprimento de fio como forma de melhorar o desempenho e diminuir o consumo do circuito. O fluxo de execução do BFG-R é mostrado na Figura 3.42.

Tabela 3.15: Resultados de *overflow* e comprimento de fio do Archer para todos os circuitos 3D do ISPD 2007.

Benchmark	Archer		
	Ovfl	WL (e5)	CPU (min)
adaptec1	0	96,16	87
adaptec2	0	97,54	23
adaptec3	0	211,82	50
adaptec4	0	194,46	12
adaptec5	0	280,56	247
newblue1	682	96,03	50
newblue2	0	143,75	7
newblue3	33394	176,63	163

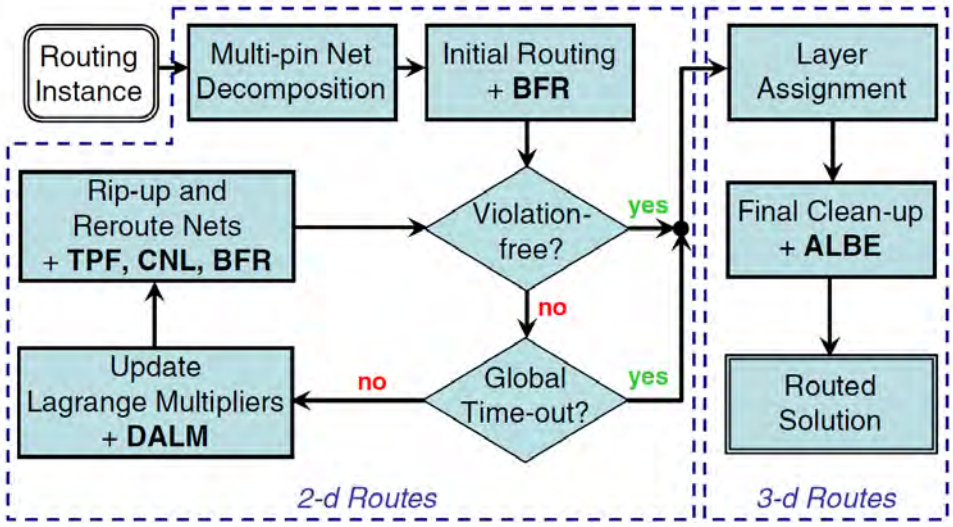

Figura 3.42: Fluxo de execução do BFG-R. (HU; ROY; MARKOV, 2010).

A decomposição das redes é feita usando MSTs, como no FGR, para maior flexibilidade. Para balancear o número de violações e o comprimento de fio, o BFG-R usa multiplicadores de Lagrange com uma função custo complementar

Para melhorar o tempo de execução, as arestas são agrupadas durante a fase de *rip-up and reroute*. As arestas com *overflow* são agrupadas com as vizinhas que também possuem *overflow*. O *rip-up and reroute* é aplicado sobre cada grupo em ordem crescente do número de *overflows*. Isso otimiza o uso da *cache* do sistema, deixando a execução mais rápida.

O ajuste dinâmico dos multiplicadores de Lagrange (sigla DALM no fluxo) é feito para que o custo histórico seja incrementado de forma diferente para arestas com muito *overflow*, para que haja um balanceamento entre o tempo de execução e a qualidade do roteamento. A atualização do custo histórico do *maze routing* é mostrada na função abaixo:

$$h_e^k = \begin{cases} h_e^{k-1} + h_{step} \times 1,25, & \text{se } e_{OF} \geq max(e_{OF}) \times 95\% \\ h_e^{k-1} + h_{step}, & \text{senão e se } e_{OF} > 0 \\ h_e^{k-1}, & \text{caso contrário} \end{cases}$$

O termo $e_{OF} \geq max(e_{OF}) \times 95\%$ engloba arestas que tem *overflow* até 5% menor que o maior *overflow*. Para o grupo com maior *overflow* as 10% arestas mais congestionadas

têm um custo adicional conforme a função abaixo:

$$h_e^k = h_e^k + (1 - cluRatio + \alpha)^{-1}$$

onde $cluRatio$ é o tamanho do grupo dividido pelo número total de arestas com *overflow* e $0 \leq \alpha < 1$ controla o crescimento da penalização, sendo $\alpha = 0,75$ um valor considerado apropriado para balancear tempo de execução e qualidade.

O termo h_{step} é ajustado dinamicamente durante o *rip-up and reroute*, entre valores mínimo e máximo (h_{step}^{min} e h_{step}^{max}). Inicialmente, $h_{step} = (h_{step}^{max} + h_{step}^{min})/2$ e o incremento $\Delta_{step} = (h_{step}^{max} - h_{step}^{max})/200$, sendo atualizado conforme a função abaixo:

$$h_e^{k+1} = \begin{cases} h_e^k + h_{step} \times 1,25, & \text{se as violações aumentaram ou não mudaram} \\ h_e^k + h_{step}, & \text{se violações diminuíram e comprimento de fio aumentou} \\ h_e^k, & \text{se violações diminuíram e comprimento de fio diminuiu} \end{cases}$$

O trabalho propõe uma modificação na função de penalização (TPF no fluxo), conforme a função abaixo:

$$p(e) = \begin{cases} \omega_e \times (1 + tg(\tau)), & \text{se } \omega > 1 \\ \omega_e, & \text{caso contrário} \end{cases}$$

onde ω_e é o *overflow* relativo e τ é o tempo relativo (tempo atual dividido pelo tempo máximo permitido). Durante o roteamento 2D, o BFG-R define o custo das vias conforme a equação abaixo: onde l é o número de camadas de roteamento disponíveis e $viaFactor$

$$p(VIA) = \lceil l/2 \rceil \times viaFactor$$

é o custo original da via 3D especificado pelo projetista.

Foi constatado que maior parte do tempo de execução era gasto no roteamento das redes com grande perímetro. Para contornar este problema, estas redes n são bloqueadas após as primeiras iterações de *rip-up and reroute* e periodicamente desbloqueadas conforme a função abaixo (sigla CNL no fluxo):

$$Period(n) = \min \left\{ \left\lceil \frac{Area(\text{BBox}_n)}{AvgArea} \right\rceil, 20 \right\}$$

onde

$$AvgArea = \left(\frac{1}{N}\right) \sum_{n=1}^{N} Area(\text{BBox}_n)$$

e

$$\begin{aligned} Area(\text{BBox}_n) \quad = \quad & (|\text{BBox}_n.x1 - \text{BBox}_n.x2| + 1) \times \\ & (|\text{BBox}_n.y1 - \text{BBox}_n.y2| + 1) \end{aligned}$$

Isso faz com que as redes grandes sejam desbloqueadas com menos frequência que as menores (mas pelo menos a cada 20 iterações) e redes na média ou menos nunca são bloqueadas

Outra técnica é estimativa de limite inferior agressiva (*aggressive lower-bound estimate* - ALBE) onde, para cada rede, em vez de usar o custo mínimo de aresta de todas as arestas para computar a sub-estimativa de distância, é percorrido o caminho a partir da última iteração de *rip-up and reroute* e usado o custo mínimo ao longo deste caminho. Como no compartilhamento do FGR, as arestas compartilhadas (que pertencem à mesma rede) têm custo menor que as não compartilhadas. Essa técnica de usar uma sub-estimativa maior não só é mais realista como também reduz o tempo de execução do A*. A ressalva a este método é que o custo pode deixar de ser sub-estimado, o que tira a admissibilidade do A* podendo, as vezes, negligenciar caminhos ótimos. Por isso, essa técnica é usada na limpeza final, onde o impacto da qualidade é negligenciável.

Outra característica do BFG-R é o uso de uma estrutura de dados que não apresenta ramificações (*branch-free representation* - BFR), como mostrado na Figura 3.43. Para cada sub-rede são armazenadas cada aresta de roteamento ocupada e as coordenadas dos seus terminais. Cada rede também armazena os índices das arestas de roteamento usadas por ela.

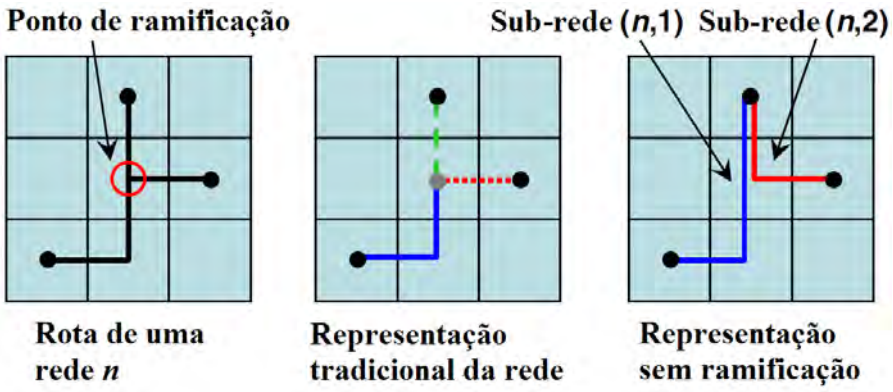

Figura 3.43: Representação sem ramificações. (HU; ROY; MARKOV, 2010).

A fase final de assinalamento de camadas é feita usando uma estratégia rápida e gulosa, seguida por uma rodada de redução de comprimento de fio 3D. Após o assinalamento das camadas, antes da "limpeza"3D (3D *clean-up*), o BFG-R aumenta temporariamente as capacidades das arestas com violações para que estas se tornem 100% utilizadas. Isso torna a solução temporariamente legal. Depois, um passo de limpeza é aplicado e encontra roteamentos mais curtos, sendo que o *overflow* máximo não pode aumentar. Após essa limpeza, as capacidades são restabelecidas. Esse método gera redução significativa de *overflow* para soluções ilegais e geralmente diminui um pouco o comprimento de fio total.

Os resultados do BFG-R para os circuitos do ISPD 2008 são mostrados na Tabela 3.16. Para melhor avaliar os roteadores, a *netlist* dos circuitos *adaptec* foram reposicionadas usando o posicionador mPl6 (CHAN et al., 2007) com as densidades de ocupação descritas na Tabela 3.17. As densidades foram escolhidas em intervalos de 10% de modo que o BFG-R conseguisse uma solução sem *overflow*. Como pode ser visto, outros roteadores não possuem soluções legais para todos os circuitos modificados.

3.3.4 NCTU-GR

O NCTU-GR (DAI; LIU; LI, 2009, 2011) é um roteador global baseado em evolução simulada, com fluxo de execução mostrado na Figura 3.44. Como outros roteadores, o

Tabela 3.16: Resultados de *overflow* e comprimento de fio do BFG-R para os circuitos do ISPD 2008.

Benchmark	BFG-R		
	Ovfl	WL (e6)	CPU (min)
adaptec1	0	5,43	8,4
adaptec2	0	5,23	3,7
adaptec3	0	13,14	16,0
adaptec4	0	12,16	5,2
adaptec5	0	15,67	15,5
bigblue1	0	5,72	10,2
bigblue2	0	9,11	40,8
bigblue3	0	13,18	20,6
bigblue4	434	23,20	1416,6
newblue1	0	4,68	256,9
newblue2	0	7,57	1,5
newblue3	33900	10,64	1420,9
newblue4	218	13,08	1413,3
newblue5	0	23,30	47,6
newblue6	0	18,01	15,7
newblue7	606	35,21	1421,1

Tabela 3.17: Resultados de *overflow* e comprimento de fio do BFG-R para os circuitos *adaptec* modificados. Comprimento de fio na escala e6 e tempo de CPU em minutos.

adaptec#	NTHU-Route 2.0			NTUgr			FastRoute 4.0			BFG-R		
	OF	WL	CPU	*OF*	WL	CPU	*OF*	WL	CPU	*OF*	WL	CPU
#1, 70%	0	4,62	7,2	0	4,83	73,2	184	5,01	26,4	0	4,68	9,8
#2, 60%	0	5,29	0,9	0	5,48	3,7	0	5,31	0,6	0	5,28	2,2
#3, 80%	38	12,16	19,4	28	12,88	470,0	616	12,74	183,1	0	12,15	27,2
#4, 80%	0	10,50	2,3	0	10,75	9,1	10	10,61	4,8	0	10,49	3,2
#5, 70%	4	13,91	25,2	0	14,44	347,8	628	14,49	50,6	0	13,98	32,6

NCTU-GR faz uso do FLUTE para roteamento inicial, sendo as redes decompostas em dois terminais por MST.

A segunda etapa do roteador tem por objetivo reduzir ao máximo o número de *overflows* antes da fase de *rip-up and reroute*. A técnica proposta é chamada de roteamento monotônico com ordenamento fixo circular (*circular fixed-ordering monotonic routing* - CFOMR). O CFOMR aplica *rip-up and reroute* repetidamente para todas as redes que estão em áreas próximas de regiões com *overflow* ou que possuem *overflow* em ordem decrescente de comprimento de fio usando roteamento monotônico k_1 vezes, com uma rede sendo desfeita e refeita por vez. Esse processo de repetição com ordem decrescente de comprimento de fio faz com que as redes sejam re-roteadas simulando uma ordenação diferente (ordenamento circular). Os estudos mostraram que a diminuição do *overflow* estabiliza para um determinado valor de k_1, o qual é definido como 5 por padrão.

O *rip-up and reroute* baseado em evolução simulada define uma pontuação para cada rede conforme mostrado abaixo. Essa pontuação é normalizada para valores entre 0,1 e 0,9. Para cada rede, um número aleatório é sorteado (entre 0 e 1) e caso a pontuação

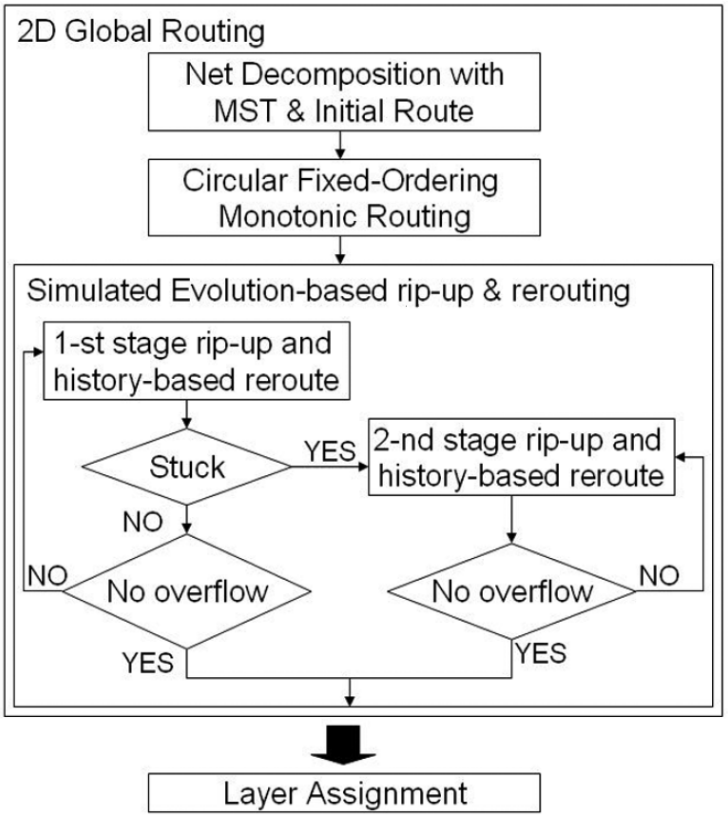

Figura 3.44: Fluxo de execução do NCTU-GR. (DAI; LIU; LI, 2011).

normalizada da rede seja maior que esse valor, esta é re-roteada.

$$pontuacao = \begin{cases} custo\_qualidade, & \text{se não há } \textit{overflow} \\ \frac{\omega}{num\_violacoes \times (k+1)} + custo\_qualidade, & \text{caso contrário} \end{cases}$$

onde k é o número de iterações, ω é um parâmetro definido pelo usuário e

$$custo\_qualidade = \beta \times (num\_vias - num\_pinos) + \gamma \times \frac{comp\_fio\_atual}{menor\_comp\_fio}$$

Com essa pontuação, redes com mais violações (*overflows*) são re-roteadas primeiro e o termo logarítmico serve para evitar que redes tenham grande crescimento de comprimento de fio caso estejam em regiões muito congestionadas, onde o número de violações não diminui durante as iterações.

O estágio de *rip-up and reroute* aplica inicialmente uma função custo baseada em histórico juntamente com *maze routing* sendo a função custo definida abaixo:

$$custo_e^k = (1 + \frac{h_e^k}{k}) \times p_e$$

onde h_e^k é o custo histórico, k o número de iterações e

$$p_e = 1 + \frac{penalizacao}{1 + e^{-inclinacao \times (demanda - capacidade)}}$$

onde a penalização e a inclinação são definidas pelo usuário.

Quando a quantidade de *overflows* estabiliza uma segunda função custo é aplicada para que a diminuição de *overflows* tenha prioridade sobre o comprimento de fio. Esta função custo é mostrada abaixo:

$$custo_e^k = (1 + (\frac{h_e^k}{log(k)+1})^\alpha) \times p_e$$

onde α é um parâmetro definido pelo usuário.

Os resultados do NCTU-GR para os circuitos do ISPD 2008 são mostrados na Tabela 3.18. O termo ajustado significa que os parâmetros definidos pelo usuário foram alterados para cada circuito, para gerar a melhor solução possível.

Tabela 3.18: Resultados de *overflow* e comprimento de fio do NCTU-GR para os circuitos do ISPD 2008.

Benchmark	NCTU-GR (ajustado)			NCTU-GR		
	Ovfl	WL (e5)	CPU (s)	*Ovfl*	WL (e5)	CPU (s)
adaptec1	0	53,40	230	0	53,50	470
adaptec2	0	51,69	87	0	51,97	121
adaptec3	0	130,08	239	0	130,08	327
adaptec4	0	120,67	132	0	120,62	634
adaptec5	0	154,7	541	0	155,01	691
bigblue1	0	55,89	378	0	58,39	497
bigblue2	0	89,4	523	0	90,01	723
bigblue3	0	129,66	261	0	130,20	322
bigblue4	164	223,99	3850	180	228,03	3344
newblue1	0	45,99	213	0	46,01	270
newblue2	0	74,86	53	0	74,80	64
newblue3	31802	104,28	6521	31872	106,22	3920
newblue4	134	126,79	2441	152	127,98	2088
newblue5	0	230,31	899	0	231,08	850
newblue6	0	176,87	553	0	180,77	720
newblue7	114	338,63	4294	144	338,68	322

3.3.5 GRVWC

O roteador proposto (HSU; CHEN; CHANG, 2008, 2010) (mesmos autores do NTUgr) insere o contexto de capacidade de vias dentro do roteamento global, para melhorar a integração com o roteamento detalhado. A capacidade de vias é calculada de acordo com a equação a seguir.

$$v_t = \left\lfloor \frac{\max\{0,\ (a_t - a_o - a_i) - a_w\}}{(v_w + v_s)^2} \right\rfloor$$

onde a_t, a_o, a_i, v_w e v_s representam a área total da GRC, a área ocupada por obstáculos, a área das redes *in-tile* (aquelas que não cruzam as fronteiras da GRC), a largura de via e o espaçamento de via, respectivamente. O termo a_w representa a provável área ocupada pelos fios que cruzam as fronteiras da GRC (metade da largura da GRC, considerando uma distribuição uniforme), de acordo com a equação abaixo:

$$a_w = \sum_{\mu \in B} (\mu_w + \mu_s) l_t / 2$$

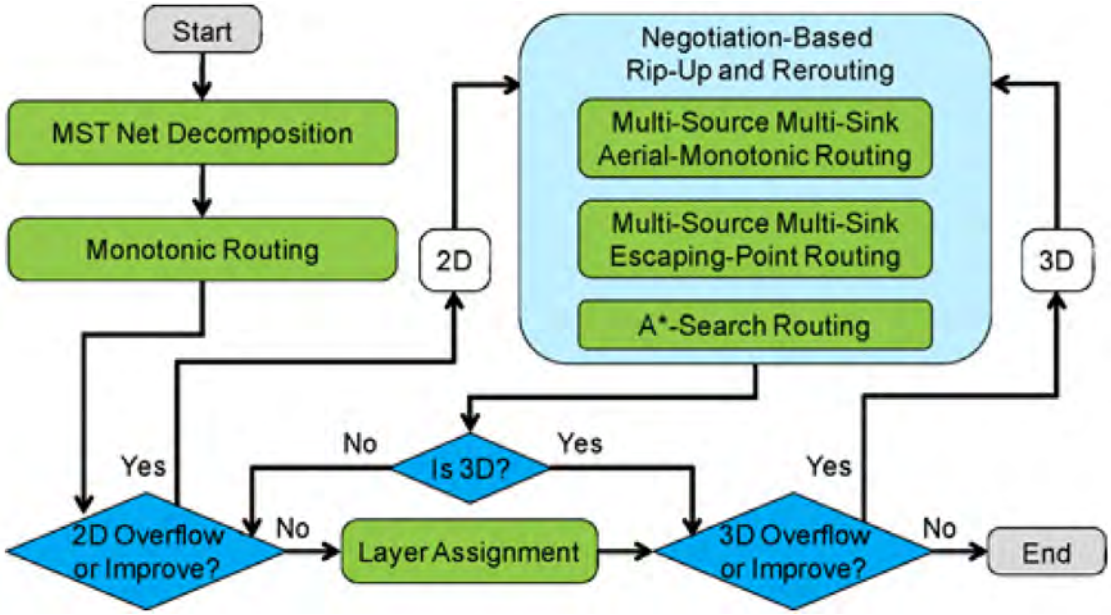

Figura 3.45: Fluxo de execução. (HSU; CHEN; CHANG, 2010).

onde l_t, u_w e u_s representam a largura da GRC, a largura de fio e o espaçamento de fio.

O fluxo do roteador pode ser visto na Figura 3.45. Como pode ser visto existem dois tratamentos diferentes no estágio principal, onde o circuito pode ser 2D ou 3D, isso porque a fase de assinalamento de camadas está inserida dentro da parte iterativa do fluxo, para que o roteador possa considerar a capacidade de vias durante o roteamento e ainda assim evitar o alto custo do roteamento 3D completo.

Na fase de *rip-up and reroute* as redes de maior perímetro são roteadas primeiro seguindo uma função custo para o roteamento 2D e outra para o roteamento 3D, mostradas abaixo, respectivamente.

$$custo_{2-D}(e) = (1 + h_e) \cdot p_e + b_e$$

$$custo_{3-D}(e) = \begin{cases} k_1 & \text{se induz } \textit{overflow} \\ (1 + h_e) \cdot p_v & \text{se induz empilhamento de via} \\ 0 & \text{se ocorre compartilhamento de fio/via} \\ 1 & \text{caso contrário} \end{cases}$$

onde h_e, p_e e b_e representam o custo histórico, de penalização (demanda/capacidade) e o custo de desvio (quando passar por e causa desvio o valor é 1, e 0 caso contrário), respectivamente. k_1 é igual a 10^6 para evitar que seja introduzido mais *overflow* e p_v a penalização de vias (demanda/capacidade).

O uso das técnicas de roteamento precisam de adaptações dos seus algoritmos para uso em roteamento 3D completo. O roteamento monotônico não têm utilidade prática em 3D, já que os pinos estão geralmente na camada mais baixa de metal. Para resolver esse problema, os autores propõem duas soluções: roteamento monotônico aéreo com

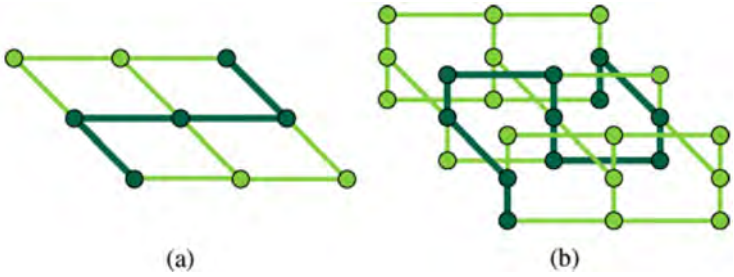

Figura 3.46: Roteamento monotônico normal e monotônico aéreo. (HSU; CHEN; CHANG, 2010).

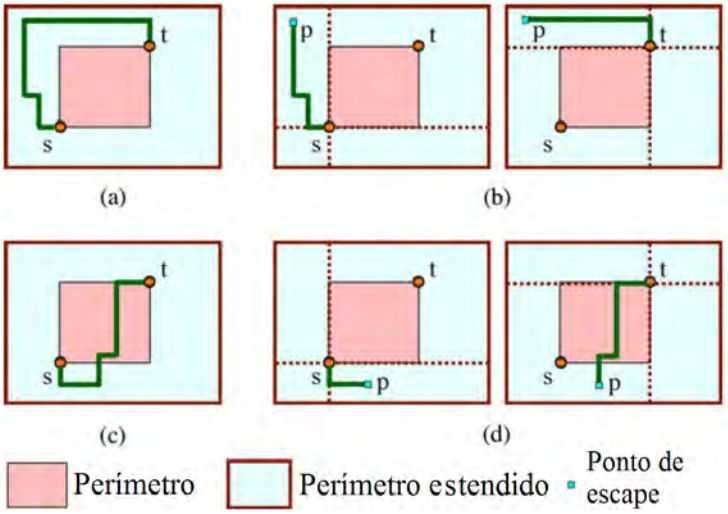

Figura 3.47: Exemplo de rede que precisa de desvio. (HSU; CHEN; CHANG, 2010).

múltiplas origens e múltiplos destinos e roteamento com múltiplas origens e múltiplos destinos com ponto de escape.

O roteamento monotônico aéreo com múltiplas origens e múltiplos destinos permite que desvios no sentido vertical sejam feitos, desde que a monotonicidade da vista aérea (de cima) do roteamento se mantenha, como mostrado na Figura 3.46.

O roteamento com múltiplas origens e múltiplos destinos com ponto de escape é uma técnica que permite que redes que não possuam roteamento monotônico sem *overflow* façam desvios. O exemplo da Figura 3.47 mostra uma situação onde a origem s e o destino t não possuem um caminho monotônico sem *overflow* e por isso o roteamento precisa sair da área dentro do perímetro da rede para ser terminado. Essa expansão da área de busca depende do congestionamento e do perímetro da rede.

A busca pelo ponto de escape é feita com a divisão da área de roteamento em quadrantes para o nodo de origem e para o de destino, como mostrado na Figura 3.48, sendo feito roteamento aéreo monotônico (propagação dos custos) até o canto oposto do nodo. Após isso, os custos são combinados e o roteamento que apresentar o menor custo é escolhido. Em resumo, o roteamento com ponto de escape é a combinação de dois roteamentos monotônicos aéreos.

A diferença do tempo de execução entre as técnicas propostas e o *maze routing* 3D, de acordo com o tamanho do perímetro expandido da rede é mostrada na Figura 3.49.

Os resultados deste roteador nos circuitos do ISPD 2008 são mostrados na Tabela 3.19. O roteador não conseguiu gerar nenhuma solução sem *overflow* de vias, de acordo

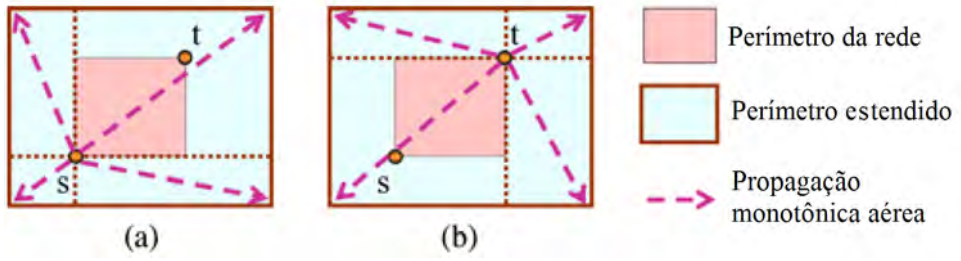

Figura 3.48: Busca do ponto de escape. (HSU; CHEN; CHANG, 2010).

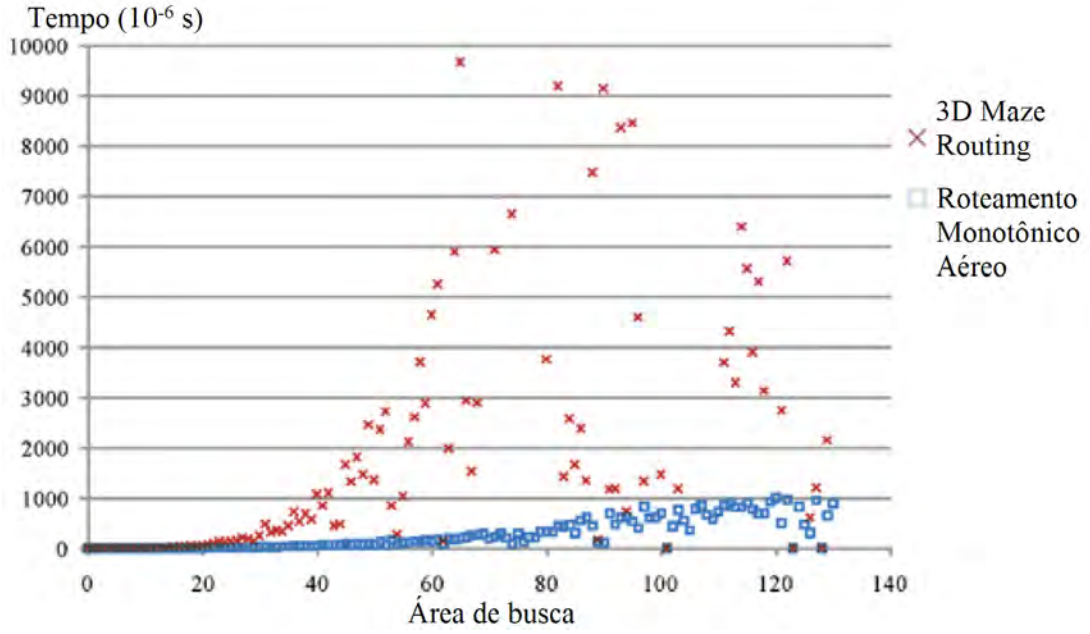

Figura 3.49: Tempo de execução de *maze routing* 3D e das técnicas propostas. Adaptada de (HSU; CHEN; CHANG, 2010).

com a métrica de capacidade de vias proposta, apresentando, entretanto, resultados com 10% menos *overflow* de vias que o NTHU-Route, para os circuitos do ISPD 2008.

3.3.6 GLADE

O GLADE (CHANG; LEE; WANG, 2010) é o primeiro roteador global desenvolvido para satisfazer as diretivas de camadas[4] propostas no ICCAD 2009 (*International Conference on Computer-Aided Design 2009*), estendendo o NTHU-Route 2.0 para tratar essas diretivas (ignorando as diretivas para comprimento máximo das redes também introduzidas no ICCAD 2009).

Para tratar as diretivas de camadas o GLADE insere uma etapa de pseudo assinalamento de camadas no NTHU, já que este faz mapeamento 3D-2D para executar o roteamento, não considerando em que camada estão os fios antes do assinalamento de camadas. Também são modificados os custos utilizados durante o roteamento e o ordenamento das redes durante o assinalamento de camadas.

O GLADE foi posteriormente estendido (LEE; CHANG; WANG, 2011) para tratar diretivas de camadas arbitrárias (as diretivas do ICCAD 2009 eram limitadas a dois tipos),

[4]Restrições que determinam em quais camadas de metal uma rede deve ser roteada.

Tabela 3.19: Resultados de *overflow* de fio e via e comprimento de fio para os circuitos 3D do ISPD 2007 e ISPD 2008. *valores medidos pela métrica do ISPD 2007.

Benchmark	GRVWC		
	Ovfl	Via *Ovfl*	WL (e5)
adaptec1	0	1081	96,61*
adaptec2	0	135459	100,68*
adaptec3	0	10390	218,76*
adaptec4	0	1616	206,14*
adaptec5	0	427068	270,81*
bigblue1	0	138362	64,04
bigblue2	0	0	89,98
bigblue3	0	28239	134,97
bigblue4	194	3780	241,49
newblue1	62	51410	95,71*
newblue2	0	4343	136,15*
newblue3	31246	-	239,29*
newblue4	148	137	140,14
newblue5	0	240	244,55
newblue6	0	186	215,40
newblue7	340	361	374,64

aplicando novas técnicas de refinamento do assinalamento de camadas já implementado. Os resultados do GLADE e dessa nova versão para os circuitos do ICCAD 2009 são mostrados na Tabela 3.20.

3.4 Discussão

A descrição detalhada dos trabalhos recentes mostra claramente que a maioria dos trabalhos possui diversos pontos em comum. A grande exceção é o fluxo proposto na ferramenta GRIP. Na maioria das ferramentas emprega-se a decomposição das redes, utilizando ferramentas anteriormente disponíveis na literatura (FLUTE na maior parte dos casos) como forma de simplificar o problema e também facilitar o uso da principal técnica de roteamento utilizada nos trabalhos, o *maze routing*.

Outro ponto em comum relevante é o uso de funções custo que captam não apenas fatores locais mas também fatores relacionados à região no entorno da aresta. O principal objetivo desta identificação de áreas congestionadas é acelerar a convergência do roteamento sem afetar a qualidade. Por se tratarem de heurísticas definidas empiricamente, a abrangência de tais técnicas torna-se duvidosa caso fossem aplicadas fora do grupo de testes definido (*benchmarks* do ISPD *Contest*). Os trabalhos também não dão detalhes sobre a efetividade dessas técnicas nos diferentes circuitos testados.

Todos os trabalhos que apresentam resultados similares aos melhores obtidos contam com etapas finais ou mesmo intermediárias de refinamento. Isso mostra que, pelo menos até então, não existe uma técnica ou método com capacidade de, sozinho, alcançar resultados compatíveis com o estado da arte, sendo necessária a aplicação de técnicas heurísticas e ajustes específicos nas técnicas para obtenção de melhores resultados num determinado conjunto de teste.

Apesar de trabalhos recentes expandirem sua capacidade para englobar novas restri-

Tabela 3.20: Resultados do GLADE e da sua versão estendida. "Vio. LD"representa as violações nas diretivas de camadas, TOF e TWL o *overflow* e o comprimento de fio totais e CPU o tempo de execução em minutos.

Circuito	GLADE				GLADE 2			
	TOF	Vio. LD	TWL	CPU	TOF	Vio. LD	TWL	CPU
adaptec1	0	0	45,4	7	0	0	45,3	10,3
adaptec2	0	0	43,9	1,4	0	0	43,8	4,2
adaptec3	0	0	115,2	7,2	0	0	114,9	11,3
adaptec4	0	0	106,5	1,8	0	0	106,5	3,9
adaptec5	0	0	130,1	15,2	0	0	129,6	26
bigblue1	0	0	48,3	8,7	0	0	48,5	17,1
bigblue2	0	0	69,6	7,8	0	0	69,1	10,4
bigblue3	0	0	105,9	3,8	0	0	105,5	10,4
bigblue4	188	0	178,9	121	188	0	177,6	324,8
newblue1	2	0	35,6	4,8	2	0	35,5	8,7
newblue2	0	0	59,7	0,8	0	0	59,6	2,4
newblue4	140	0	108,1	40,1	140	0	107,7	48,6
newblue5	0	0	190,7	12,6	0	0	190,3	20,8
newblue6	0	0	139,8	11,5	0	0	139	23,7
newblue7	78	0	281,7	119,9	78	0	279,3	169,9
Rel.	-	1,000	1,000	1,000	-	1,000	0,996	1,904

ções e características de projetos modernos, o núcleo das principais ferramentas continua baseado nas mesmas técnicas e simplificações já apresentadas anteriormente, com destaque para decomposição de redes, negociação e *maze routing*.

Como citado no início desta seção, o trabalho que apresenta um grande diferencial na abordagem do problema é o GRIP. Apesar de usar ILP como diversos trabalhos anteriormente publicados, a mudança da formulação proposta foi capaz de gerar resultados excelentes e muito melhores dos obtidos pelos trabalhos anteriores usando ILP. Como todos os outros trabalhos, o GRIP não possui um método global suficientemente eficiente para não precisar de etapas finais de refinamento. Além disso, o trabalho ainda conta com o fator negativo do grande tempo de execução quando comparado com os demais trabalhos com similar qualidade de resultados.

4 ROTEADOR GLOBAL DESENVOLVIDO

Dentro do contexto de roteamento global exposto nos capítulos anteriores e pela necessidade de desenvolvimento de uma ferramenta de roteamento global dentro do grupo de pesquisa, este trabalho propõe um fluxo capaz de tratar o roteamento global de circuitos VLSI atuais (com tamanhos da ordem de milhões de redes) apresentando técnicas novas e outras já encontradas na literatura apresentada.

Por se tratar de uma primeira implementação, o objetivo deste trabalho é explorar ao máximo as características do roteamento global em circuitos atuais, inicialmente analisando técnicas que não envolvam a identificação de áreas nos circuitos, ou seja, neste trabalho não há qualquer técnica que diferencie regiões do circuito de acordo com características de congestionamento ou similar como encontrado em todos os outros trabalhos publicados após as competições do ISPD 2007 e 2008 e os participantes desta última.

Neste capítulo são descritos os detalhes das técnicas utilizadas e os resultados alcançados para os circuitos de *benchmark* disponíveis, com uma comparação frente aos trabalhos anteriores e as técnicas utilizadas nestes.

4.1 Fluxo do Roteador Global (GR)

O fluxo de execução do GR é apresentado na Figura 4.1. As etapas presentes no fluxo e a estrutura de dados utilizada são detalhadas nas seções a seguir. Neste trabalho foram desenvolvidas duas versões da ferramenta. Uma das versões tem por objetivo ter o menor comprimento de fio possível e a outra busca a convergência da solução o mais rápido possível, ou seja, com o menor número possível de iterações, diminuindo o tempo de execução. Estas versões serão chamadas ao longo do texto de "Versão WL"e "Versão RT", respectivamente. Além dessa diferenciação de versões também foram estudados os resultados alterando-se a ferramenta de construção das árvores das redes dos circuitos, entre MST e SMT (FLUTE).

4.2 Estrutura de Dados

A estrutura de dados é basicamente dividida em dois grupos principais: a lista de todas as redes e seus respectivos fios e a representação da grade de roteamento. A grade de roteamento é representada como uma matriz bidimensional, onde cada elemento representa uma GRC. Cada GRC possui associada a ela quatro arestas, denominadas intuitivamente norte, sul, leste e oeste, como mostrado na Figura 4.3. As arestas que representam a borda do circuito possuem capacidade zero, não podendo ser usadas durante o roteamento.

Em cada GRC são mantidas as suas coordenadas, as referências para as arestas e um

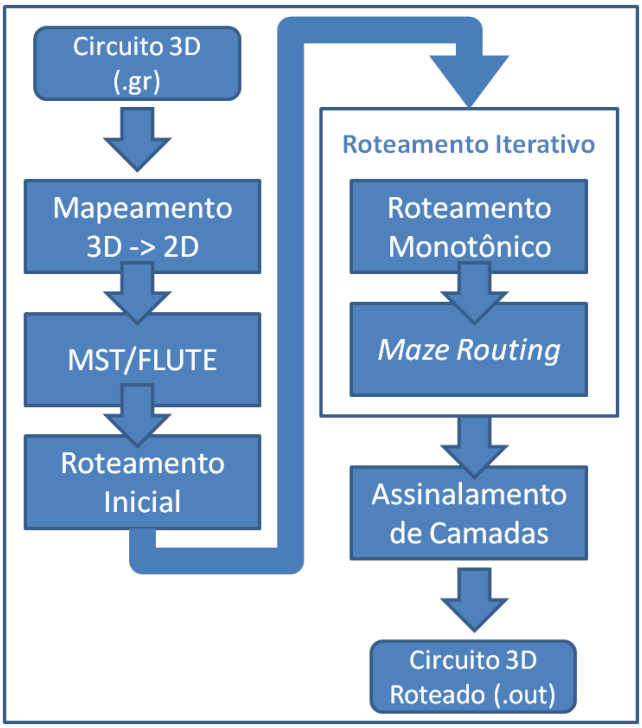

Figura 4.1: Fluxo de execução do GR.

Figura 4.2: Representação gráfica da estrutura de uma GRC.

campo para o retraço do *maze routing*. Já nas arestas ficam todas as informações referentes aos fios e redes que passam através delas e as características da aresta como capacidade, custos e direção (vertical ou horizontal).

Os fios são armazenados na forma de um vetor, sendo que cada fio mantém a informação a qual rede pertence. Deste modo toda a rede pode ser acessada (e desfeita e/ou refeita) através da informação guardada no próprio fio.

4.3 Construção e Decomposição das Redes

Na implementação deste trabalho existem duas formas de construção das redes do circuito: *minimum spanning trees* (MSTs) e *Steiner minimum trees* (SMTs). A construção por MST é feita pelo algoritmo RMST-Pack (ANDREW B. KAHNG, ????) e a construção por SMT é feita pelo FLUTE. Após a construção das árvores para cada rede, esta é decomposta em redes de dois terminais (fios), caso necessário.

O uso de MSTs dá mais flexibilidade ao roteador global, já que não há nodos de *Steiner*. Este método gera árvores com comprimento de fio maior do que as geradas por SMT. Entretanto, como o *maze routing* do roteador identifica o compartilhamento das arestas por diferentes fios da mesma rede, a solução gerada pode conter diversas árvores

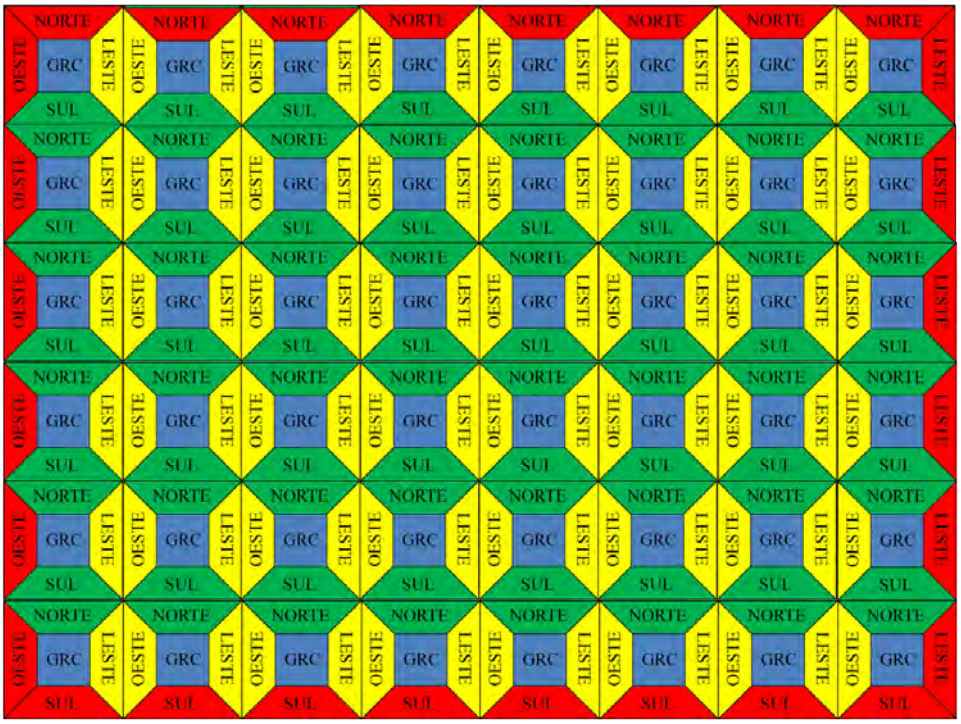

Figura 4.3: Exemplo da estrutura da grade de roteamento. Arestas em vermelho representam as bordas do circuito, ou seja, tem capacidade zero.

de *Steiner* construídas ao longo das iterações.

Os caminhos dos fios são representados pela lista de arestas pelas quais ele passa, ou seja, não há uma representação dos segmentos deste fio. Na grade de roteamento, em cada aresta, é mantida uma lista com todos os fios que passam por esta aresta, sendo construída a partir desta lista a lista com todas as redes que passam por esta aresta, o que na prática determina o número real de fios que passam pela aresta, ou seja, sua demanda.

4.4 Roteamento Inicial

Nesta etapa são roteados apenas os fios com comprimento menor (limitados a duas unidades de comprimento). Este roteamento é feito apenas com formas "L"para os fios que não são unidirecionais, ou seja, que não podem ser roteados com apenas uma linha reta. A presença desta etapa se deve ao fato de este tipo de roteamento ser muito ágil quando comparada ao *maze routing*, tornando a primeira iteração mais rápida sem degradar a qualidade da solução. O ordenamento dos fios é baseado na área ocupada pela rede a qual o fio pertence, começando pela menor até a maior rede.

Para a Versão WL o roteamento inicial é executado para as redes com comprimento de fio de até duas unidades e para a Versão RT são roteadas todos fios. Essa escolha foi feita para dar mais agilidade à primeira iteração da Versão RT, uma vez que por ter no máximo duas possibilidades de roteamento (formas "L") este tipo de roteamento não é capaz de identificar possíveis arestas de compartilhamento (e portanto redução de comprimento de fio). Isto faz com que redes que não estão em áreas congestionadas, ou seja, que não serão desfeitas posteriormente, possam não ter a melhor solução possível em termos de comprimento de fio, sendo então apenas os fios menores roteados por esta etapa na versão WL.

Na Figura 4.4 é mostrado o mapa de congestionamento do circuito adaptec2 antes e depois do roteamento inicial para a Versão WL e na Figura 4.5 para a Versão RT.

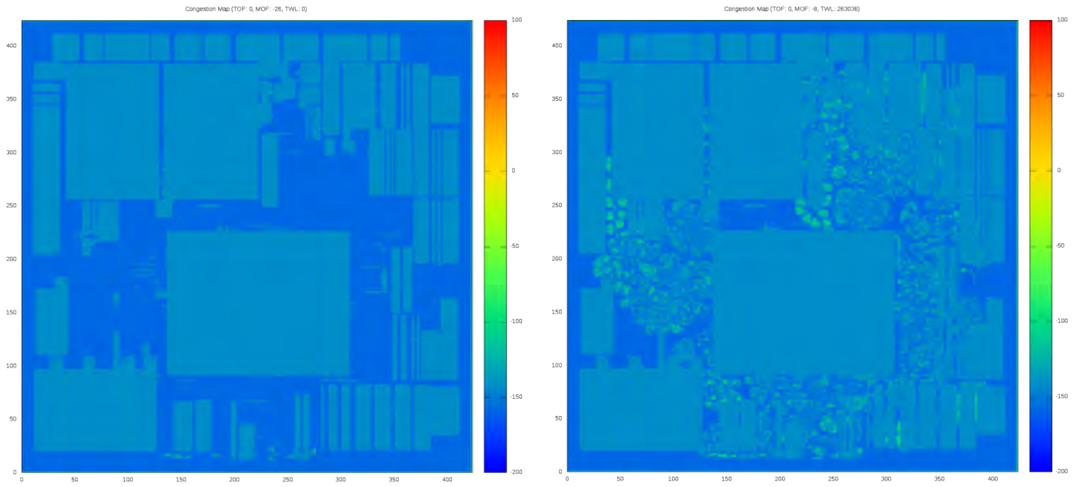

Figura 4.4: Mapa de congestionamento do circuito adaptec2 antes e depois do roteamento inicial (Versão WL).

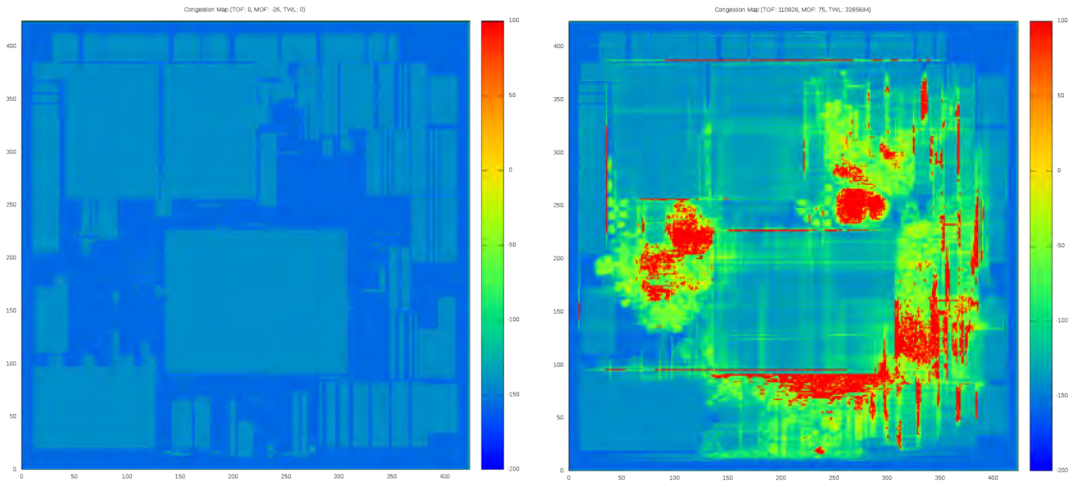

Figura 4.5: Mapa de congestionamento do circuito adaptec2 antes e depois do roteamento inicial (Versão RT). Regiões em vermelho indicam GRCs onde há *overflow*

Na versão WL o roteamento inicial é responsável pelo roteamento de 20% a 60% dos fios, de acordo com o circuito. Já na versão RT, entre 75% e 98% dos fios são roteados nesta fase. Na identificação de congestionamento posterior a esta etapa, o custo histórico não é incrementado.

4.5 Roteamento Iterativo Baseado em Negociação

Esta etapa do roteamento compreende o uso de duas técnicas de roteamento: roteamento monotônico e *maze routing*. Estas duas etapas são detalhadas nas seções seguintes. A utilização destas duas técnicas difere nas duas versões da ferramenta, não só no momento da sua utilização mas também nos parâmetros utilizados nestas técnicas. Em ambas as versões não é permitido ao roteamento monotônico criar violações, ou seja, o roteamento monotônico apenas gera uma solução para um determinado fio se esta solução não cria *overflow* em nenhuma aresta. Além disso, na versão RT o roteamento monotônico além de não poder criar violações também não pode ocupar arestas com mais de 95% de sua capacidade já ocupada por outros fios (a não ser que nesta aresta já passe outro fio da mesma rede). Isso faz com que o roteamento monotônico seja mais rápido (já que desistirá mais rápido caso suas arestas iniciais já não tenham a capacidade disponível necessária) e também fique restrito a fios com solução mais simples. Na versão WL o roteamento monotônico é desabilitado caso o *overflow* total seja inferior a 2,5% do *overflow* inicial, e na versão RT quando o *overflow* total seja inferior a 15% do *overflow* inicial. Esse critério mais agressivo para a versão RT é para que o número de iterações diminua, já que o roteamento monotônico tem muito menos liberdade que o *maze routing*, atrasando a convergência da solução.

O *maze routing* é utilizado de duas formas durante o roteamento, uma limitada (como o roteamento monotônico) e outra sem o critério de capacidade da aresta (apenas custo). Na forma limitada o *maze routing* apenas gera uma solução para o fio caso não encontre *overflow* em nenhuma aresta do seu caminho. Isto acelera a convergência da solução (aumentando o comprimento de fio) mas também desacelera as iterações, já que para os fios que não possuam essa solução sem *overflow* será feito o *maze routing* duas vezes (limitado e não limitado) e também esse *maze routing* limitado pode ser mais demorado que o original por expandir sobre mais GRCs caso o terminal de destino esteja bloqueado em todas direções. No *maze routing* normal, apenas o custo das arestas é considerado, não havendo restrição quanto a capacidade das arestas. O *maze routing* limitado apenas é habilitado quando: (1) o *overflow* médio por aresta é inferior a 1,25 na versão WL e 2,5 na RT, (2) o *overflow* total é inferior a 5% do inicial na versão WL e 15% na RT e (3) caso o *overflow* tenha reduzido menos de 20% na última iteração na versão WL e menos que 25% na versão RT. Uma vez alcançado estes critérios o *maze routing* limitado é utilizado em todas iterações subsequentes[1]. Ambas as formas de *maze routing* tem a área de expansão limitada, de acordo com a equação abaixo.

$$maze\_area = min(\zeta, 1.01 + \frac{\eta}{min_{ov} + 1} + min(2, \frac{TOF^i}{TOF^{i-1}} \times \frac{e^{\epsilon \times i}}{10}))$$

onde i é a iteração atual, TOF^i o *overflow* total da iteração atual, TOF^{i-1} o *overflow* total da iteração anterior e ζ, η e ι definidos na Tabela 4.1 para as duas versões implementadas.

Tabela 4.1: Definição das constantes da área de expansão do *maze routing*.

Constante	Versão WL	Versão RT
ζ	5	10
η	200	500
ϵ	0,15	0,20

[1]Em alguns circuitos o *maze routing limitado não chega a ser habilitado*

Este valor multiplica o comprimento de fio atual para definir a distância máxima (do destino mais a distância da origem) que um nodo expandido pode ter, ou seja, caso essa multiplicação resulte num valor duas unidades superior ao comprimento de fio mínimo, o *maze routing* poderá expandir um nodo para fora da área entre os dois pinos do fio. Para garantir que fios curtos conseguiam expandir além da área mínima, após a multiplicação é somado o valor constante 2 ao resultado, permitindo que o *maze routing* possa expandir no mínimo um nodo para fora da área calculada.

O primeiro termo garante que possa-se aumentar no mínimo 1% o comprimento de fio atual, o segundo termo aumenta a liberdade do *maze routing* nas iterações finais e o termo final (limitado ao valor máximo 2) aumenta a liberdade do *maze routing* ao longo das iterações desde que o *overflow* total tenha diminuído pouco na última iteração.

Cada iteração do roteamento iterativo inicia com a identificação de quais fios serão desfeitos e re-roteados. Todos os fios que passam por pelo menos uma aresta com *overflow* são selecionados para serem re-roteados. Quando um fio selecionado é o primeiro da fila de prioridade para ser re-roteado, não apenas este é desfeito mas também todos os outros fios da sua rede são desfeitos. Isto evita que o compartilhamento das arestas por diferentes fios da mesma rede impeça que estes fios desviem destas arestas. Este problema pode ocorrer porque cada fio é roteado separadamente, assim, fios que compartilham arestas sempre enxergarão estas arestas com custo próximo de zero e caso as únicas arestas com *overflow* pelas quais estes fios passem forem estas sendo compartilhadas, o roteamento destes fios nunca desviaria do congestionamento, obrigando outros fios que passam por estas arestas a desviar delas, criando uma preferência indesejada para estes fios que compartilham arestas. Os fios da mesma rede a serem re-roteados são ordenados crescentemente de acordo com a área definida entre seus pinos[2], já que fios menores tem menos liberdade para procurar arestas compartilhadas. Cada fio re-roteado é devidamente desmarcado para não ser re-roteado na mesma iteração, evitando que, caso mais de um fio da mesma rede possua *overflow*, estes sejam re-roteados mais de uma vez na mesma iteração.

Visando agilizar as primeiras iterações, onde muitos fios (e consequentemente redes) precisam ser re-roteados, cada fio, antes de ser re-roteado, é checado pra verificar se ainda está passando por arestas congestionadas. Esta checagem é feita até que o *overflow* médio das arestas seja superior a 1.1. Esta técnica torna mais importante o ordenamento dos fios, já que apenas os primeiro fios da fila desviarão do caminho anterior. A aplicação desta técnica resulta em uma diminuição do número de fios roteados na faixa de 10% a 60%.

O ordenamento dos fios é feito com base em quatro critérios. O primeiro critério é o congestionamento médio das arestas pelas quais este fio passa. Este valor é calculado somando-se todas as diferenças entre demanda e capacidade das arestas pelas quais o fio passa e dividindo pelo número de arestas pelas quais o fio passa. Quanto menor este valor maior a prioridade do fio, ou seja, será re-roteado antes. Este critério foi definido para que os fios que possuam mais chances de desviar de áreas congestionadas sejam re-roteados primeiro, liberando recursos para os fios que estão dentro dessas regiões congestionada (e portanto terão uma média de congestionamento maior). Os outros três critérios servem como desempate do critério inicial, respectivamente na ordem na qual são descritos. O segundo critério é a divisão entre o comprimento atual do fio e seu comprimento mínimo, ou seja, o quanto o fio já desviou do seu caminho inicial, tendo prioridade os que mais desviaram. O terceiro critério é a área entre os pinos do fio, tendo prioridade as maiores

[2] A área nunca é nula, mesmo em fios onde uma de suas coordenadas seja igual nos dois pinos, sendo o lado mínimo do retângulo unitário

áreas. O último critério é o comprimento mínimo do fio, ou seja a distância entre os dois pinos, tendo os fios maiores a prioridade.

A última característica diferenciada do roteamento iterativo implementado foi desenvolvida para ajudar a convergência nas últimas iterações, quando a negociação precisaria de muitas iterações para achar a solução sem *overflows*. Neste momento a evolução do custo histórico ocorre de forma muito lenta, já que são poucos fios e muitas possibilidades de roteamento para estes, assim os fios precisam de muitas iterações para encontrar o caminho por onde outros fios poderão desviar das arestas congestionadas[3]. Para diminuir esse número de iterações (e consequentemente o tempo de execução), além de serem selecionados para o re-roteamento os fios nas arestas com *overflow*, também são selecionados os fios que passam pelas arestas com maior custo histórico. Essas arestas são selecionadas desde que tenham custo histórico superior a, inicialmente, 99% do maior custo histórico encontrado na iteração anterior e demanda uma unidade inferior à capacidade da aresta. A porcentagem é reduzida em 1% (multiplicada por 0,99) a cada iteração posterior a iteração em que este critério foi habilitado (três iterações consecutivas sem redução no *overflow* total). Isto aumenta o número de fios a serem re-roteados, assim foi estabelecido como limite 2% do total de fios do circuito como critério de parada para a queda na porcentagem, o que, ao acontecer retorna-se à porcentagem da iteração anterior. Caso novamente passem três iterações sem a redução do *overflow* total a queda na porcentagem passa a ser de 0,5%. Essa combinação faz com que o número de fios a serem re-roteados, nessa situação, nunca diminua e se aproxime progressivamente de 2% do número total de fios. Por fim, caso o *overflow* total não diminua de uma iteração para a outra, o incremento do custo histórico (h_{inc} na seção 4.5.2) é aumentado em 5%.

Ambas as técnicas de roteamento consideram o custo da via com valor constante igual a 3. Os critérios de parada do roteamento iterativo são: (1) caso encontre uma solução sem violações, (2) caso o *overflow* total não diminua por 20 iterações consecutivas[4] e (3) caso o limite de tempo de execução (24 horas) esteja próximo (é necessário deixar tempo para o assinalamento de camadas).

4.5.1 Roteamento Monotônico

O roteamento monotônico é uma forma simplificada do *maze routing*, onde apenas os nodos vizinhos que estão mais próximos do destino podem ser expandidos. Assim neste tipo de roteamento o comprimento de fio não aumenta (se forem desconsideradas as vias). Na Figura 4.6 pode-se ver como seria uma expansão a partir de uma origem (ponto no canto inferior esquerdo) até um determinado destino (canto superior direito).

Este procedimento agiliza a execução de cada iteração, entretanto não permite que os fios desviem de áreas com maior congestionamento. Por isso, durante a execução de cada iteração, o roteamento monotônico só é usado até certo ponto do roteamento iterativo (principalmente nas primeiras iterações) e de forma a só gerar uma solução caso esta não apresente *overflow*.

As funções de custo utilizadas para o roteamento monotônico são as mesmas utilizadas no *maze routing* e descritas a seguir.

[3]Este problema foi relatado no artigo do FGR e denominado *last-gasp problem*, entretanto os autores não citaram as possíveis causas do problema.

[4]Este critério é ignorado caso o *overflow* total seja menor que 20.

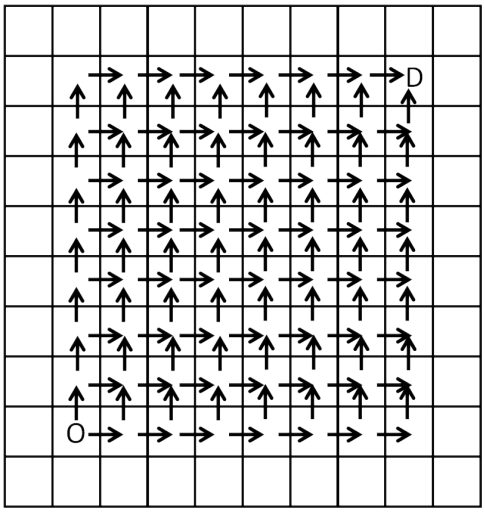

Figura 4.6: Expansão do roteamento monotônico.

4.5.2 *Maze Routing*

O *maze routing* implementado usa a técnica A*, onde a distância do nodo sendo pesquisado (em relação a origem e ao destino) é somada ao custo. Esta técnica permite que o número de nodos expandido seja menor, indo em direção ao destino e evitando expansões desnecessárias. Na Figura 4.7 vemos um possível exemplo de expansão do *maze routing* combinando o custo da distância (A*) com o custo dos nodos e o custo das vias (onde há mudança de direção).

Para ajudar a remediar o problema da convergência quando o *overflow é pequeno*, citado anteriormente, o custo de penalização cresce de acordo com a diminuição do número total de *overflows*, ou seja, quanto menor o número de violações maior é o custo nas arestas onde há *overflow*. Assim a função custo utiliza é a mostrada abaixo.

$$custo_e = b_e + h_e \times p_e$$

onde b_e é o custo base (igual a 1), h_e é o custo histórico e p_e o custo de penalização atual. Os custos h_e e p_e são definidos abaixo.

$$h_e^i = \begin{cases} h^{i-1}, & \text{se } d_e < c_e \\ h^{i-1} + k \times h_{inc}, & \text{caso contrário} \end{cases}$$

sendo que $k = 1$ e h_{inc} é igual a 1,25 caso a diferença entre demanda e capacidade seja superior ou igual a 95% da máxima violação encontrada, e 1 caso contrário.

$$p_e = c_0 \times e^{((i \times \alpha + \frac{\beta}{(min_{ov}+10)} + \gamma) \times (d_e - c_e + 1))}$$

onde i é a iteração atual, min_{ov} é o mínimo *overflow* total já encontrado e c_0, α, β e γ são constantes definidas pelo usuário (na implementação atual apenas antes da compilação) e que possuem valores diferentes nas duas versões definidas neste trabalho, especificadas na Tabela 4.2. Estes valores foram definidos inicialmente com a observação da curva gerada e depois foram levemente alterados de acordo com os resultados experimentais[5].

[5]Por questões de precisão e estouro de representação númerica na implementacão, o custo de penalização é limitado a um valor máximo constante de 10000.

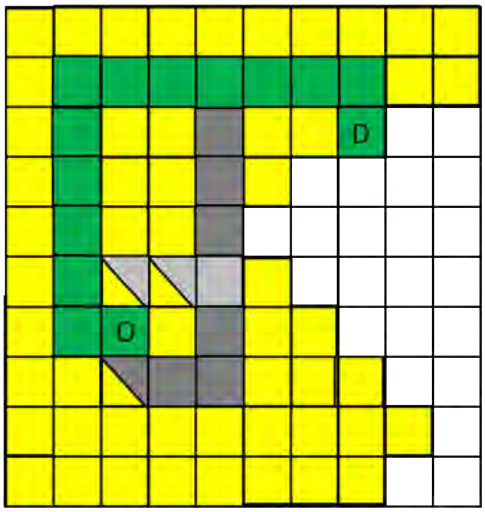

Figura 4.7: Exemplo de possível expansão do *maze routing* considerando custos dos nodos e custos de vias (dobras). Áreas em cinza mais escuro representam custos de nodos maiores. Nodos amarelos seriam os nodos expandidos mas não utilizados no caminho final e em verde o caminho escolhido.

Como explicado na seção 4.2 os custos são armazenados nas próprias arestas, assim para evitar que os novos custos tenham que ser calculados a cada avaliação de custo durante o roteamento iterativo (inserção na fila de prioridade), os custos já são calculados com um deslocamento unitário (no expoente de p_e), ou seja, como se a demanda já fosse uma unidade maior que a demanda atual real, acontecendo o mesmo com o incremento do custo histórico, como mostrado na equação deste. A constante c_0 determina inicialmente qual o custo em uma aresta onde a capacidade e a demanda serão iguais caso um fio seja adicionado a esta aresta. Quanto maior este valor mais fios serão deslocados mesmo que a aresta não esteja congestionada, o que faz o comprimento de fio aumentar mais rapidamente bem como o roteamento iterativo convergir mais rapidamente.

Na Figura 4.8 é mostrado o mapa de congestionamento do circuito adaptec2 após o roteamento iterativo ter encontrado uma solução sem violações.

Tabela 4.2: Definição das constantes do custo de penalização para as duas versões da ferramenta.

Constante	Versão WL	Versão RT
c_0	1,0	5,0
α	0,015	0,020
β	50	50
γ	0,30	0,35

4.6 Assinalamento de Camadas

A etapa final do roteamento é o assinalamento de camadas dos fios (remapeamento para 3D). Nesta etapa foi implementado um algoritmo guloso, sem uma função objetivo, com a única restrição de que não haja mudanças na solução 2D gerada, ou seja, o número de violações deve permanecer igual. O algoritmo guloso é capaz de encontrar essa solução pois não há restrição com relação ao número de vias dentro de uma GRC.

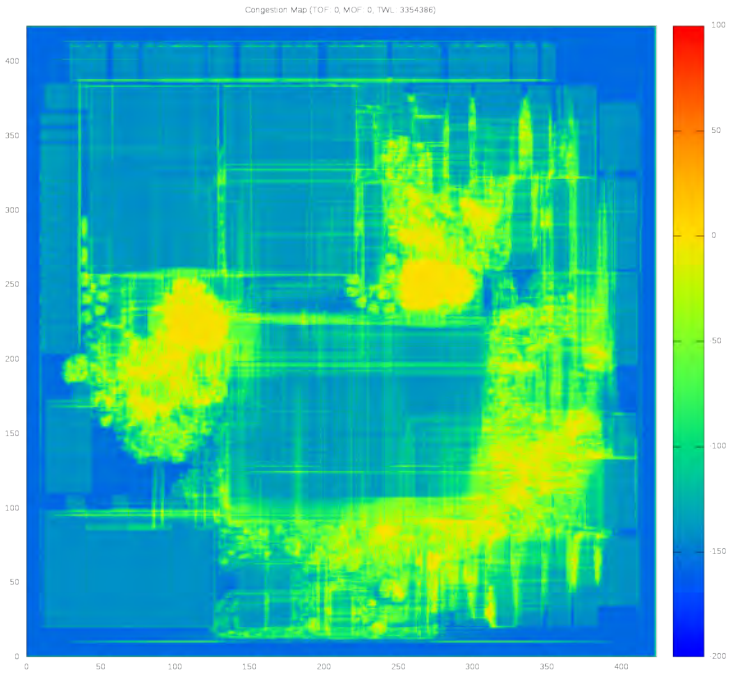

Figura 4.8: Circuito adaptec2 após o roteamento iterativo ter encontrado uma solução sem violações.

Também neste caso o ordenamento das redes tem grande influência na solução gerada. Na implementação as redes são assinaladas em ordem crescente de área. Esse ordenamento evita que redes de área menor, ou seja, com conexões mais curtas, tenham que ir para camadas superiores de metal, sendo que as redes com maior área tem uma probabilidade maior de, em algum ponto do seu roteamento ter que ir para as camadas superiores.

4.7 Resultados Experimentais

A ferramenta foi desenvolvida em C++ e testada com os *benchmarks* do ISPD 1998 e do ISPD 2008 (que incluem os circuitos 3D do ISPD 2007). Os experimentos foram realizados em uma máquina com oito processadores Intel(R) Xeon(R) E5520 @2.27GHz, com 6GB de memória RAM (nenhum tipo de paralelismo foi utilizado na ferramenta). A ferramenta foi compilada utilizando o GCC 4.4.3.

4.7.1 Resultados para Circuitos do ISPD 1998

A primeira vez reportada na literatura em que os circuitos do ISPD 1998 foram todos roteados sem violações foi na publicação do FGR 1.0 (ROY; MARKOV, 2007). Isto evidencia o grande salto de qualidade observado nos roteadores globais acadêmicos com o acontecimento dos concursos do ISPD 2007 e 2008. Na Tabela 4.3 vemos um resumo das características destes circuitos.

Para estes circuitos a ferramenta desenvolvida sempre utiliza o FLUTE para geração das árvores de roteamento. Assim, existem apenas duas versões da ferramenta, a que

Tabela 4.3: Características dos circuitos do ISPD 1998. *menor comprimento de fio possível calculado usando o GeoSteiner (CHO et al., 2007).

Circuito	Redes	Grade	Cap. Vert.	Cap. Hor.	Menor WL*
ibm01	11507	64 × 64	12	14	60142
ibm02	18429	80 × 64	22	34	165863
ibm03	21621	80 × 64	20	30	145678
ibm04	26163	96 × 64	20	23	162734
ibm06	33354	128 × 64	20	33	275868
ibm07	44394	192 × 64	21	36	363537
ibm08	47944	192 × 64	21	32	402412
ibm09	50393	256 × 64	14	28	411260
ibm10	64227	256 × 64	27	40	574407

Tabela 4.4: Resultados de *overflow* e comprimento de fio do roteador implementado para os circuitos do ISPD 1998. *valores obtidos em ROY; MARKOV (2007). **valores sem o assinalamento de camadas.

Circuito	FGR*			GR - WL			GR - RT		
	TOF	WL	CPU (s)	TOF	WL	CPU (s)**	TOF	WL	CPU (s)**
ibm01	0	63332	10,0	0	63520	2,5	0	65385	1,3
ibm02	0	168918	13,0	0	170384	5,5	0	174921	3,1
ibm03	0	146412	5,0	0	146772	3,5	0	149047	2,2
ibm04	0	167101	29,0	0	169773	18,6	0	172732	17,0
ibm06	0	277608	18,0	0	278835	10,1	0	283693	5,5
ibm07	0	366180	20,0	0	367737	12,1	0	372320	7,1
ibm08	0	404714	18,0	0	405182	11,8	0	409694	7,0
ibm09	0	413053	20,0	0	414988	10,1	0	419979	6,9
ibm10	0	578795	92,0	0	581149	31,5	0	590660	13,9

busca otimizar o comprimento de fio e a que busca menor tempo de execução, WL e RT, respectivamente. Da mesma forma que todos roteadores globais atuais, a ferramenta gera solução livre de violações em todos os circuitos[6], sendo que para a versão WL o tempo total para rotear os nove circuitos é de 152 segundos e para a versão RT é de 116 segundos. No caso destes circuitos, a ferramenta não necessitaria da fase final de assinalamento de camadas como a necessária para os *benchmarks* novos, já que são circuitos 2D, por esta razão os tempos reportados não incluem o tempo gasto no assinalamento de camadas.

Na Tabela 4.4 são mostrados os tempos de execução e comprimentos de fio para todos os circuitos e os mesmos obtidos pelo FGR 1.0. Os resultados apresentam valores muito similares, tanto em tempo de execução (já que o FGR foi executado em uma máquina mais antiga) quanto na qualidade das soluções. Aqui vale ressaltar que em nenhum momento a ferramenta desenvolvida teve como objetivo os circuitos aqui apresentados, podendo haver parâmetros (constantes de custos, constantes de decisão, etc.) que ajustados, resultariam em comprimentos de fio menores.

[6]os circuitos foram obtidos no formato de entrada do ISPD 2007 no site do BoxRouter, sendo que o circuito ibm05 não está disponível por se tratar de um caso trivial, ou seja, possui solução sem violações apenas com roteamento de formas "L".

4.7.2 Resultados para Circuitos do ISPD 2008

A Tabela 4.5 apresenta o número de redes e tamanho da grade de roteamento dos circuitos do ISPD 2008. A Tabela 4.6 mostra os resultados da ferramenta implementada usando construção das árvores de roteamento por MST, com as informações de *overflow* total (TOF), maior *overflow* existente (MOF), comprimento de fio total sem custo de via (WL) e tempo total de execução da ferramenta (CPU). Já a Tabela 4.7 mostra os mesmos resultados para construção das árvores de roteamento pelo FLUTE. Em ambas o custo das vias é ignorado, sendo mostradas ambas versões da ferramenta e a diferença percentual entre elas.

Fica evidente que o modelamento de custos utilizado na versão RT gera um comprimento de fio maior, de aproximadamente 1,7% em média para as duas formas de construção de árvores de roteamento. Em termos de tempo de execução da ferramenta, não há um comportamento igual para todos os circuitos. Nos circuitos em que a ferramenta consegue gerar solução livre de violações o tempo de execução é em média 33% menor na versão RT (MST) sendo que os circuitos bigblue2, bigblue4, newblue1 e newblue4 há um aumento de execução, para os quais o critério de parada se torna menos eficiente, já que a abordagem mais agressiva da versão RT faz com que a solução demore mais a estabilizar, resultado do aumento desnecessário do comprimento de fio nas iterações iniciais.

Tabela 4.5: Número de redes, camadas e grade dos circuitos do ISPD 2008. Min. WL representa o menor comprimento de fio total possível de acordo com o FLUTE.

Circuito	Redes	Grade	Camadas	Min. WL
adaptec1	176715	324×324	6	3385957
adaptec2	207972	424×424	6	3201709
adaptec3	368494	774×779	6	9333399
adaptec4	401060	774×779	6	8874536
adaptec5	548073	465×468	6	9789557
bigblue1	196885	227×227	6	3426311
bigblue2	428968	468×471	6	4594408
bigblue3	665629	555×557	8	7683689
bigblue4	1133535	403×405	8	11582044
newblue1	270713	399×399	6	2321710
newblue2	373790	557×463	6	4598245
newblue3	442005	973×1256	6	7375987
newblue4	631292	455×458	6	7947677
newblue5	891920	637×640	6	14191356
newblue6	835267	463×464	6	9743154
newblue7	1647410	488×490	8	17747262

Como pode ser visto o uso do FLUTE não necessariamente garante uma diminuição do comprimento de fio total. Isso se deve ao fato de não haver re-estruturação da rede durante a negociação, ou seja, os nodos de Steiner comportam-se como pinos da rede e não podem ser movidos ou apagados (o que não afetaria a validade da solução). Esse problema não só dificulta a negociação, como pode ser visto pelos tempos de execução, mas também pode tornar um circuito não roteável, como vemos no caso do circuito newblue5, para o qual a ferramenta consegue gerar uma solução sem violações usando construção de árvores de roteamento por MST mas apresenta um valor elevado de violações (1576) usando o FLUTE.

Tabela 4.6: Resultados de *overflow* e comprimento de fio do roteador implementado usando MST para os circuitos do ISPD 2008. TOF é o *overflow* total. WL é o comprimento de fio sem custos de via. Tempo de CPU em minutos.

MST	GR - WL				GR - RT (Δ)			
Circ.	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU
a1	0	0	3672608	46,2	0	0	3728681 (+1,5%)	27,7 (-40%)
a2	0	0	3354386	15,3	0	0	3409108 (+1,6%)	7,2 (-53%)
a3	0	0	9677595	45,4	0	0	9886302 (+2,2%)	20,4 (-55%)
a4	0	0	9026091	9,4	0	0	9100198 (+0,8%)	6,9 (-27%)
a5	0	0	10452584	120,9	0	0	10635696 (+1,8%)	74,1 (-39%)
b1	0	0	3826593	165,9	0	0	3965458 (+3,6%)	114,9 (-31%)
b2	122	2	5237687	281,3	130	2	5331616 (+1,8%)	303,8 (8%)
b3	0	0	7940693	56,3	0	0	7985105 (+0,6%)	44,6 (-21%)
b4	780	6	12075198	197,1	588	16	12315097 (+2,0%)	421,7 (+114%)
n1	588	2	2486311	89,8	580	2	2504647 (+0,7%)	143,9 (+60%)
n2	0	0	4681756	5,4	0	0	4721392 (+0,9%)	3,6(-33%)
n3	36292	1194	7683682	1416,8	34602	1200	7834083 (2,0%)	1380,8 (-3%)
n4	462	2	8567326	323,5	400	2	8754043 (+2,2%)	328,8 (+2%)
n5	0	0	15045883	283,8	0	0	15249504 (+1,4%)	168,9 (-41%)
n6	0	0	10420794	164,6	0	0	10661630 (+2,3%)	74,1 (-55%)
n7	5172	4	18617528	974,2	6154	6	18904697 (+1,5%)	967,8 (-1%)

As Tabelas 4.8, 4.9, 4.10 e 4.11 mostram os resultados dos roteadores com código disponível (FastRoute, NTUgr e NTHU) e o resultado relativo médio (última linha da tabela)[7]. Observando as tabelas fica claro a grande diferença quando é considerado o número de vias. Os resultados obtidos sem considerar o número de vias mostram que o roteador implementado apresenta qualidade compatível com as ferramentas do estado da arte no que diz respeito à qualidade de roteamento (comprimento de fio), com a diferença média não ultrapassando 1,80% (em relação ao NTHU), mesmo considerando os circuitos onde o roteador não encontra solução sem violações, onde o roteador desenvolvido apresenta as maiores diferença. Sem considerar tais circuitos, a diferença máxima é de 1,56% (também em relação ao NTHU).

[7]Todos os valores de diferença porcentual apresentados neste capítulo usam a ferramenta desenvolvida como referência.

Tabela 4.7: Resultados de *overflow* e comprimento de fio do roteador implementado usando FLUTE para os circuitos do ISPD 2008. TOF é o *overflow* total. WL é o comprimento de fio sem custos de via. Tempo de CPU em minutos.

FLUTE	GR - WL				GR - RT (Δ)			
Circ.	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU
a1	0	0	3683810	65,9	0	0	3711829 (+0,8%)	35,7 (-46%)
a2	0	0	3346596	30,7	0	0	3418557 (+2,2%)	32,8 (+7%)
a3	0	0	9605438	44,7	0	0	9797434 (+2,0%)	20,1 (-55%)
a4	0	0	8932222	9,5	0	0	8970127 (+0,4%)	7,0 (-26%)
a5	0	0	10430261	161,2	0	0	10599781 (+1,6%)	83,3 (-48%)
b1	0	0	3836507	368,5	0	0	3978022 (+3,7%)	239,4 (-35%)
b2	776	4	5089263	392,9	596	8	5154019 (+1,3%)	461,8 (+18%)
b3	0	0	7951793	312,5	0	0	8058265 (+1,3%)	136,3 (-56%)
b4	1546	30	12053813	306,5	1316	32	12251184 (+1,6%)	286,7 (-6%)
n1	1766	4	2422752	120,5	1810	4	2456155 (+1,4%)	104,0 (-14%)
n2	0	0	4628547	5,0	0	0	4647910 (+0,4%)	3,7 (-26%)
n3	35242	1230	7666336	876,9	33292	1222	7799741 (+1,7%)	1078,5 (+23%)
n4	1456	4	8321997	247,0	842	22	8573346 (+3,0%)	1135,5 (+360%)
n5	1576	8	14839333	612,8	1186	14	15272408 (+2,9%)	1099,1 (+80%)
n6	0	0	10443840	216,7	0	0	10665844 (+2,1%)	101,3 (-53%)
n7	6888	6	18511502	803,4	8014	6	18772417 (+1,4%)	861,8 (+7%)

Tabela 4.8: Resultados dos roteadores globais com código disponível e roteador implementado usando MST, para custo de via zero. Tempo de CPU em minutos. *O NTHU estourou o tempo limite de 24 horas na iteração 56, sendo os valores mostrados informados pela própria ferramenta. Custo de via nulo.

	FastRoute				NTUgr				NTHU				GR - WL			
Circ.	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU
a1	0	0	3619358	6,6	0	0	3715732	46,8	0	0	3591612	5,4	0	0	3672608	46,2
a2	0	0	3323509	1,7	0	0	3411210	7,6	0	0	3305724	2,4	0	0	3354386	15,3
a3	0	0	9660062	7,4	0	0	9879292	37,8	0	0	9659956	6,0	0	0	9677595	45,4
a4	0	0	8992578	1,6	0	0	9119619	14,3	0	0	8957270	1,7	0	0	9026091	9,4
a5	0	0	10380076	12,5	0	0	10699866	110,9	0	0	10297697	14,4	0	0	10452584	120,9
b1	0	0	3787895	6,3	0	0	3903183	136,2	0	0	3710213	10,7	0	0	3826593	165,9
b2	130	4	4941000	18,8	118	4	5013784	215,9	86	2	4808514	10,9	122	2	5237687	281,3
b3	0	0	7888855	4,1	0	0	8037384	29,9	32	4	7826368	7,6	0	0	7940693	56,3
b4	172	4	12120977	64,3	410	10	12478456	313,1	256	2	11983118	67,6	780	6	12075198	197,1
n1	42	4	2430402	12,6	212	4	2491986	157,5	164	2	2393336	14,0	588	2	2486311	89,8
n2	0	0	4668763	1,1	0	0	4753628	4,3	0	0	4635947	0,8	0	0	4681756	5,4
n3	31826	726	7652336	67,3	33636	-	0	183,1	39832*	1088*	7845673*	1440,0*	36292	1194	7683682	1416,8
n4	214	4	8293985	43,2	284	4	8517969	254,5	222	2	8229937	31,1	462	2	8567326	323,5
n5	0	0	14881030	5,8	0	0	15296919	117,9	18	2	14692577	23,0	0	0	15045883	283,8
n6	0	0	10377887	16,5	0	0	10561501	72,6	0	0	10239595	28,8	0	0	10420794	164,6
n7	628	6	18518498	564,2	906	6	0	1421,0	68	2	18922284	0,0	5172	4	18617528	974,2
Rel.			0,988				1,012				0,982				1,000	

Tabela 4.9: Resultados dos roteadores globais com código disponível e roteador implementado usando FLUTE, para custo de via zero. Tempo de CPU em minutos. *O NTHU estourou o tempo limite de 24 horas na iteração 56, sendo os valores mostrados informados pela própria ferramenta. Custo de via nulo.

	FastRoute				NTUgr				NTHU				GR - WL			
Circ.	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU
a1	0	0	3619358	6,6	0	0	3715732	46,8	0	0	3591612	5,4	0	0	3683810	65,9
a2	0	0	3323509	1,7	0	0	3411210	7,6	0	0	3305724	2,4	0	0	3346596	30,7
a3	0	0	9660062	7,4	0	0	9879292	37,8	0	0	9659956	6,0	0	0	9605438	44,7
a4	0	0	8992578	1,6	0	0	9119619	14,3	0	0	8957270	1,7	0	0	8932222	9,5
a5	0	0	10380076	12,5	0	0	10699866	110,9	0	0	10297697	14,4	0	0	10430261	161,2
b1	0	0	3787895	6,3	0	0	3903183	136,2	0	0	3710213	10,7	0	0	3836507	368,5
b2	130	4	4941000	18,8	118	4	5013784	215,9	86	2	4808514	10,9	776	4	5089263	392,9
b3	0	0	7888855	4,1	0	0	8037384	29,9	32	4	7826368	7,6	0	0	7951793	312,5
b4	172	4	12120977	64,3	410	10	12478456	313,1	256	2	11983118	67,6	1546	30	12053813	306,5
n1	42	4	2430402	12,6	212	4	2491986	157,5	164	2	2393336	14,0	1766	4	2422752	120,5
n2	0	0	4668763	1,1	0	0	4753628	4,3	0	0	4635947	0,8	0	0	4628547	5,0
n3	31826	726	7652336	67,3	33636	374	7785823	183,1	39832*	1088*	7845673*	1440,0*	35242	1230	7666336	876,9
n4	214	4	8293985	43,2	284	4	8517969	254,5	222	2	8229937	31,1	1456	4	8321997	247,0
n5	0	0	14881030	5,8	0	0	15296919	117,9	18	2	14692577	23,0	1576	8	14839333	612,8
n6	0	0	10377887	16,5	0	0	10561501	72,6	0	0	10239595	28,8	0	0	10443840	216,7
n7	628	6	18518498	564,2	906	6	19290419	1421,0	68	2	18922284	0,0	6888	6	18511502	803,4
Rel.			0,996				1,021				0,990				1,000	

Tabela 4.10: Resultados dos roteadores globais com código disponível e roteador implementado usando MST, para custo de via zero. Tempo de CPU em minutos. *O NTHU estourou o tempo limite de 24 horas na iteração 56, sendo os valores mostrados informados pela própria ferramenta. Custo de via igual a um.

	FastRoute				NTUgr				NTHU				GR - WL			
Circ.	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU
a1	0	0	5339159	6,6	0	0	5609943	46,8	0	0	5358337	5,4	0	0	6380538	46,2
a2	0	0	5196541	1,7	0	0	5421602	7,6	0	0	5235189	2,4	0	0	6084400	15,3
a3	0	0	13059584	7,4	0	0	13649616	37,8	0	0	13151201	6,0	0	0	15155118	45,4
a4	0	0	12111169	1,6	0	0	12408382	14,3	0	0	12175553	1,7	0	0	13981761	9,4
a5	0	0	15669576	12,5	0	0	16358988	110,9	0	0	15535086	14,4	0	0	18102395	120,9
b1	0	0	5761898	6,3	0	0	5949946	136,2	0	0	5572807	10,7	0	0	6685407	165,9
b2	130	4	9493390	18,8	118	4	9467229	215,9	86	2	8995514	10,9	122	2	11204002	281,3
b3	0	0	12983114	4,1	0	0	13491167	29,9	32	4	13072116	7,6	0	0	15594684	56,3
b4	172	4	23388053	64,3	410	10	24049722	313,1	256	2	22800848	67,6	780	6	27602383	197,1
n1	42	4	4748422	12,6	212	4	4824320	157,5	164	2	4601467	14,0	588	2	5437199	89,8
n2	0	0	7497894	1,1	0	0	7769446	4,3	0	0	7587705	0,8	0	0	9245356	5,4
n3	31826	726	10786866	67,3	33636	374	11005863	183,1	39832*	1088*	-	1440,0*	36292	1194	13316366	1416,8
n4	214	4	13203198	43,2	284	4	13442722	254,5	222	2	12885658	31,1	462	2	15329779	323,5
n5	0	0	23221921	5,8	0	0	24021512	117,9	18	2	23138793	23,0	0	0	26983599	283,8
n6	0	0	17929554	16,5	0	0	18735225	72,6	0	0	17690104	28,7	0	0	20708074	164,6
n7	628	6	35566479	564,1	906	6	37485508	1421,0	68	2	35524829	0,0	5172	4	42224489	974,2
Rel.			0,851				0,879				0,844				1,000	

Tabela 4.11: Resultados dos roteadores globais com código disponível e roteador implementado usando FLUTE, para custo de via zero. Tempo de CPU em minutos. *O NTHU estourou o tempo limite de 24 horas na iteração 56, sendo os valores mostrados informados pela própria ferramenta. Custo de via igual a um.

	FastRoute				NTUgr				NTHU				GR - WL			
Circ.	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU	TOF	MOF	WL	CPU
a1	0	0	5339159	6,6	0	0	5609943	46,8	0	0	5358337	5,4	0	0	6371496	65,9
a2	0	0	5196541	1,7	0	0	5421602	7,6	0	0	5235189	2,4	0	0	6054036	30,7
a3	0	0	13059584	7,4	0	0	13649616	37,8	0	0	13151201	6,0	0	0	15023535	44,7
a4	0	0	12111169	1,6	0	0	12408382	14,3	0	0	12175553	1,7	0	0	13791352	9,5
a5	0	0	15669576	12,5	0	0	16358988	110,9	0	0	15535086	14,4	0	0	18007166	161,2
b1	0	0	5761898	6,3	0	0	5949946	136,2	0	0	5572807	10,7	0	0	6674218	368,5
b2	130	4	9493390	18,8	118	4	9467229	215,9	86	2	8995514	10,9	776	4	10946954	392,9
b3	0	0	12983114	4,1	0	0	13491167	29,9	32	4	13072116	7,6	0	0	15538857	312,5
b4	172	4	23388053	64,3	410	10	24049722	313,1	256	2	22800848	67,6	1546	30	27315327	306,5
n1	42	4	4748422	12,6	212	4	4824320	157,5	164	2	4601467	14,0	1766	4	5309688	120,5
n2	0	0	7497894	1,1	0	0	7769446	4,3	0	0	7587705	0,8	0	0	9123742	5,0
n3	31826	726	10786866	67,3	33636	374	11005863	183,1	39832*	1088*	-	1440,0	35242	1230	13059064	876,9
n4	214	4	13203198	43,2	284	4	13442722	254,5	222	2	12885658	31,1	1456	4	14905590	247,0
n5	0	0	23221921	5,8	0	0	24021512	117,9	18	2	23138793	23,0	1576	8	26563051	612,8
n6	0	0	17929554	16,5	0	0	18735225	72,6	0	0	17690104	28,7	0	0	20702529	216,7
n7	628	6	35566479	564,1	906	6	37485508	1421,0	68	2	35524829	0,0	6888	6	41821161	803,35
Rel.			0,860				0,888				0,853				1,000	

Na Tabela 4.12 são mostrados os resultados do roteador global desenvolvido e do GRIP[8] para custo de via um. A diferença de comprimento de fio é de em média 18,67%, o que mostra o impacto de uma boa etapa de assinalamento de camadas, já que os resultados sem considerar o custo das vias é de 0,66% em média (Tabela 4.12).

Tabela 4.12: Resultados de *overflow* e comprimento de fio do roteador desenvolvido e do GRIP, com custo de vias.

MST	GRIP				GR - WL			
Circ.	TOF	MOF	WL	CPU(min)	TOF	MOF	WL	CPU(min)
a1	0	0	5173917	388	0	0	6380538	46,18
a2	0	0	5032264	455	0	0	6084400	15,33
a3	0	0	12717508	478	0	0	15155118	45,42
a4	0	0	11872636	509	0	0	13981761	9,38
a5	0	0	15035901	584	0	0	18102395	120,88
b1	0	0	5400347	339	0	0	6685407	165,90
b2	0	0	8651290	690	122	2	11204002	281,30
b3	0	0	12647077	731	0	0	15594684	56,25
b4	176	22	22214356	726	780	6	27410378	197,05
n1	0	0	4437266	483	588	2	5437199	89,75
n2	0	0	7227839	467	0	0	9245356	5,37
n3	47812	636	10279005	1430	36292	1194	13134074	1416,80
n4	132	16	12438956	529	462	2	15329779	323,45
n5	0	0	22376245	821	0	0	26983599	283,83
n6	0	0	17197951	448	0	0	20708074	164,60
n7	54	18	33859392	985	5172	4	42139409	974,20
Rel.			0,813				1,000	

[8]O GRIP apresenta os melhores resultados de comprimento de fio pela métrica do ISPD 2008.

Tabela 4.13: Resultados de *overflow* e comprimento de fio do roteador desenvolvido e do GRIP, sem custo de vias.

MST	GRIP				GR - WL			
Circ.	TOF	MOF	WL	CPU(min)	TOF	MOF	WL	CPU(min)
a1	0	0	3647798	388	0	0	3672608	46,2
a2	0	0	3377578	455	0	0	3354386	15,3
a3	0	0	9752798	478	0	0	9677595	45,4
a4	0	0	9149524	509	0	0	9026091	9,4
a5	0	0	10481522	584	0	0	10452584	120,9
b1	0	0	3727006	339	0	0	3826593	165,9
b2	0	0	4836489	690	122	2	5237687	281,3
b3	0	0	7867123	731	0	0	7940693	56,3
b4	176	22	12189701	726	780	6	12075198	197,1
n1	0	0	2491899	483	588	2	2486311	89,8
n2	0	0	4804545	467	0	0	4681756	5,4
n3	47812	636	7603392	1430	36292	1194	7683682	1416,8
n4	132	16	8329288	529	462	2	8567326	323,5
n5	0	0	14773105	821	0	0	15045883	283,8
n6	0	0	10246769	448	0	0	10420794	164,6
n7	54	18	18946837	985	5172	4	18617528	974,2
Rel.			0,993				1,000	

5 CONCLUSÃO

Este trabalho apresenta a implementação de um roteador global capaz de tratar os problemas de roteamento atuais. Os resultados mostraram que a ferramenta já é capaz de gerar resultados com boa qualidade mesmo sem fazer uso de técnicas de identificação de áreas de congestionamento, sem otimizações pós-roteamento e sem nenhuma forma de ajuste (*tuning*) para os diferentes circuitos de *benchmark*.

O foco durante o processo de desenvolvimento e implementação da ferramenta foram os circuitos mais recentes, entretanto a ferramenta demonstrou grande eficiência para resolver os problemas mais antigos (ISPD 1998), gerando soluções com qualidade similar ou melhor que os principais roteadores encontrados na literatura.

A ferramenta apresenta uma diferença em relação aos melhores resultados já reportados na literatura para circuitos 3D lançados no ISPD 2008 de, em média, 1,78% na métrica de comprimento de fio sem considerar o custo das vias e de 15,56% considerando o custo da via como uma unidade de comprimento de fio (ISPD 2008), para a versão voltada a otimização de comprimento de fio. Já para a versão da ferramenta que busca a convergência o mais rápido possível a diferença foi de 3,39% (custo de via igual a 0) e 16,32% (custo de via igual a 1). Nota-se que as maiores diferenças são encontradas nos circuitos mais difíceis de gerar uma solução sem violações. Isso mostra como as técnicas de identificação de região podem contribuir tanto para uma convergência mais rápida quanto para evitar que fios tomem rotas desnecessárias durante a fase de negociação. Comparando-se com os melhores resultados encontrados na literatura (GRIP) tem-se em média 18,67% maior comprimento de fio para custo de via unitário.

Considerando o tempo de execução da ferramenta, fica claro que esta ainda necessita melhorias de implementação para a etapa de roteamento iterativo. Por outro lado, o uso de memória está adequado (e em alguns casos muito melhor), quando comparado aos roteadores com código disponível e que puderam ser avaliados.

A partir deste trabalho também poderão ser desenvolvidos diversos estudos e futuras melhorias na implementação, principalmente para melhorar seu tempo de execução. Neste contexto, os trabalhos futuros mais iminentes são o desenvolvimento e implementação de um algoritmo de assinalamento de camadas mais robusto e que inclua a otimização do número de vias e outros objetivos, a identificação dos nodos de Steiner para que o desempenho da ferramenta melhore quando esta utiliza o FLUTE para a construção das árvores de roteamento a adequação da ferramenta para os parâmetros propostos nos *benchmarks* do ICCAD 2009 (*International Conference on Computer-Aided Design*), estudar a adequação da ferramenta para que se torne dirigida a desempenho (com a construção de árvores de roteamento dirigidas a desempenho) e inserir a ferramenta dentro do fluxo de projeto ASTRAN (ZIESEMER; LAZZARI, 2007; POSSER et al., 2010; REIS, 2011).

REFERÊNCIAS

ALBRECHT, C. Global routing by new approximation algorithms for multicommodity flow. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.20, n.5, p.622 –632, may 2001.

ANDREW B. KAHNG, I. M. **RMST-Pack - Rectilinear Minimum Spanning Tree Algorithms**. ., Disponível em: `http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/RMST/`. Acesso em: 14/12/2012.

BOESE, K. et al. Near-optimal critical sink routing tree constructions. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.14, n.12, p.1417 –1436, dec 1995.

BORAH, M.; OWENS, R.; IRWIN, M. An edge-based heuristic for Steiner routing. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.13, n.12, p.1563 –1568, dec 1994.

BURSTEIN, M.; PELAVIN, R. Hierarchical Wire Routing. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.2, n.4, p.223 – 234, Oct. 1983.

CARDEN R.C., I.; LI, J.; CHENG, C.-K. A global router with a theoretical bound on the optimal solution. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.15, n.2, p.208 –216, feb 1996.

CHAN, T. et al. mPL6: enhanced multilevel mixed-size placement with congestion control. In: CHANDRAKASAN, A.; NAM, G.-J.; CONG, J. (Ed.). **Modern Circuit Placement**. [S.l.]: Springer US, 2007. p.247–288. (Series on Integrated Circuits and Systems).

CHANG, Y.-J. et al. NTHU-Route 2.0: A Robust Global Router for Modern Designs. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.29, n.12, p.1931 –1944, Dec. 2010.

CHANG, Y.-J.; LEE, T.-H.; WANG, T.-C. GLADE: a modern global router considering layer directives. In: COMPUTER-AIDED DESIGN (ICCAD), 2010 IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.319 –323.

CHANG, Y.-J.; LEE, Y.-T.; WANG, T.-C. NTHU-Route 2.0: A fast and stable global router. In: COMPUTER-AIDED DESIGN, 2008. ICCAD 2008. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.338 –343.

CHEN, H.-Y.; HSU, C.-H.; CHANG, Y.-W. High-performance global routing with fast overflow reduction. In: DESIGN AUTOMATION CONFERENCE, 2009. ASP-DAC 2009. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2009. p.582 –587.

CHEN, Y.-A.; LIN, Y.-L.; HSU, Y.-C. A new global router for ASIC design based on simulated evolution. In: VLSI TECHNOLOGY, SYSTEMS AND APPLICATIONS, 1989. PROCEEDINGS OF TECHNICAL PAPERS. 1989 INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 1989. p.261 –265.

CHIANG, C.; SARRAFZADEH, M.; WONG, C. Global routing based on Steiner min-max trees. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.9, n.12, p.1318 –1325, dec 1990.

CHIANG, C.; WONG, C.; SARRAFZADEH, M. A weighted Steiner tree-based global router with simultaneous length and density minimization. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.13, n.12, p.1461 – 1469, dec 1994.

CHO, J.; SARRAFZADEH, M. Four-bend top-down global routing. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.17, n.9, p.793 –802, sep 1998.

CHO, M. et al. BoxRouter 2.0: architecture and implementation of a hybrid and robust global router. In: COMPUTER-AIDED DESIGN, 2007. ICCAD 2007. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.503 –508.

CHO, M. et al. BoxRouter 2.0: a hybrid and robust global router with layer assignment for routability. **ACM Trans. Des. Autom. Electron. Syst.**, New York, NY, USA, v.14, p.32:1–32:21, April 2009.

CHO, M.; PAN, D. BoxRouter: a new global router based on box expansion and progressive ilp. In: DESIGN AUTOMATION CONFERENCE, 2006 43RD ACM/IEEE. **Anais...** [S.l.: s.n.], 2006. p.373 –378.

CHO, M.; PAN, D. BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.26, n.12, p.2130 –2143, dec 2007.

CHU, C.; WONG, Y.-C. FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.27, n.1, p.70 –83, jan 2008.

CONG, J.; KOH, C.-K. Interconnect layout optimization under higher-order RLC model. In: COMPUTER-AIDED DESIGN, 1997. DIGEST OF TECHNICAL PAPERS., 1997 IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1997. p.713 –720.

CONG, J.; LEUNG, K.-S.; ZHOU, D. Performance-Driven Interconnect Design Based on Distributed RC Delay Model. In: DESIGN AUTOMATION, 1993. 30TH CONFERENCE ON. **Anais...** [S.l.: s.n.], 1993. p.606 – 611.

CONG, J.; MADDEN, P. H. Performance driven global routing for standard cell design. In: THE 1997 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 1997. p.73–80. (ISPD '97).

DAI, K.-R.; LIU, W.-H.; LI, Y.-L. Efficient simulated evolution based rerouting and congestion-relaxed layer assignment on 3-D global routing. In: DESIGN AUTOMATION CONFERENCE, 2009. ASP-DAC 2009. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2009. p.570 –575.

DAI, K.-R.; LIU, W.-H.; LI, Y.-L. NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-d global routing. **Very Large Scale Integration (VLSI) Systems, IEEE Transactions on**, [S.l.], v.PP, n.99, p.1 –14, 2011.

DEES JR., W. A.; KARGER, P. G. Automated rip-up and reroute techniques. In: THE 19TH DESIGN AUTOMATION CONFERENCE, Piscataway, NJ, USA. **Anais...** IEEE Press, 1982. p.432–439. (DAC '82).

ESBENSEN, H. A macro-cell global router based on two genetic algorithms. In: THE CONFERENCE ON EUROPEAN DESIGN AUTOMATION, Los Alamitos, CA, USA. **Anais...** IEEE Computer Society Press, 1994. p.428–433. (EURO-DAC '94).

GAO, J.-R.; WU, P.-C.; WANG, T.-C. A new global router for modern designs. In: DESIGN AUTOMATION CONFERENCE, 2008. ASPDAC 2008. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2008. p.232 –237.

HADSELL, R.; MADDEN, P. Improved global routing through congestion estimation. In: DESIGN AUTOMATION CONFERENCE, 2003. PROCEEDINGS. **Anais...** [S.l.: s.n.], 2003. p.28 – 31.

HANAN, M. On Steiner's Problem with Rectilinear Distance. **SIAM Journal on Applied Mathematics**, [S.l.], n.14, 1966.

HART, P.; NILSSON, N.; RAPHAEL, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. **Systems Science and Cybernetics, IEEE Transactions on**, [S.l.], v.4, n.2, p.100 –107, July 1968.

HASAN, N.; LIU, C. L. A force-directed global router. In: IN ADVANCED RESEARCH IN VLSI. **Anais...** The MIT Press, 1987. p.135–150.

HAYASHI, M.; TSUKIYAMA, S. A hybrid hierarchical approach for multi-layer global routing. In: EUROPEAN DESIGN AND TEST CONFERENCE, 1995. ED TC 1995, PROCEEDINGS. **Anais...** [S.l.: s.n.], 1995. p.492 –496.

HONG, X. et al. TIGER: an efficient timing-driven global router for gate array and standard cell layout design. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.16, n.11, p.1323 –1331, nov 1997.

HOU, H.; HU, J.; SAPATNEKAR, S. Non-Hanan routing. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.18, n.4, p.436 –444, apr 1999.

HSU, C.-H.; CHEN, H.-Y.; CHANG, Y.-W. Multi-layer global routing considering via and wire capacities. In: COMPUTER-AIDED DESIGN, 2008. ICCAD 2008. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.350 –355.

HSU, C.-H.; CHEN, H.-Y.; CHANG, Y.-W. Multilayer Global Routing With Via and Wire Capacity Considerations. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.29, n.5, p.685 –696, may 2010.

HU, J.; ROY, J. A.; MARKOV, I. L. Sidewinder: a scalable ilp-based router. In: THE 2008 INTERNATIONAL WORKSHOP ON SYSTEM LEVEL INTERCONNECT PREDICTION, New York, NY, USA. **Anais...** ACM, 2008. p.73–80. (SLIP '08).

HU, J.; ROY, J. A.; MARKOV, I. L. Completing high-quality global routes. In: THE 19TH INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 2010. p.35–41. (ISPD '10).

HU, J.; SAPATNEKAR, S. Algorithms for non-Hanan-based optimization for VLSI interconnect under a higher-order AWE model. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.19, n.4, p.446 –458, Apr. 2000.

HU, J.; SAPATNEKAR, S. A timing-constrained algorithm for simultaneous global routing of multiple nets. In: COMPUTER AIDED DESIGN, 2000. ICCAD-2000. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2000. p.99 –103.

HU, J.; SAPATNEKAR, S. S. A Survey on Multi-Net Global Routing for Integrated Circuits. **Integration, the VLSI Journal**, [S.l.], v.31, p.1–49, 2001.

HU, T. C.; SHING, M. T. A decomposition algorithm for circuit routing. In: VLSI CIRCUIT LAYOUT: THEORY AND DESIGN. **Anais...** IEEE Press, 1985. p.144–152.

HUANG, J. et al. An Efficient Timing-Driven Global Routing Algorithm. In: DESIGN AUTOMATION, 1993. 30TH CONFERENCE ON. **Anais...** [S.l.: s.n.], 1993. p.596 – 600.

JING, T. et al. A min-area solution to performance and RLC crosstalk driven global routing problem. In: DESIGN AUTOMATION CONFERENCE, 2005. PROCEEDINGS OF THE ASP-DAC 2005. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2005. v.1, p.115 – 120 Vol. 1.

JOHANN, M. d. O. **Novos algoritmos para roteamento de circuitos VLSI**. 2001. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação.

JOHANN, M. O. et al. A New Bidirectional Heuristic Shortest Path Search Algorithm. In: INTERNATIONAL ICSC CONGRESS ON ARTIFICIAL INTELLIGENCE AND APPLICATIONS. **Anais...** [S.l.: s.n.], 2000.

JOHANN, M. O. et al. Admissibility Proofs for the LCS* Algorithm. In: INTERNATIONAL JOINT CONFERENCE, 7TH IBERO-AMERICAN CONFERENCE ON AI: Advances in Artificial Intelligence. **Proceedings...** [S.l.: s.n.], 2000. p.236–244.

KAHNG, A.; MANDOIU, I.; ZELIKOVSKY, A. Highly scalable algorithms for rectilinear and octilinear Steiner trees. In: DESIGN AUTOMATION CONFERENCE, 2003. PROCEEDINGS OF THE ASP-DAC 2003. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2003. p.827 – 833.

KASTNER, R.; BOZORGZADEH, E.; SARRAFZADEH, M. Predictable routing. In: COMPUTER AIDED DESIGN, 2000. ICCAD-2000. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2000. p.110 –113.

KASTNER, R.; BOZORGZADEH, E.; SARRAFZADEH, M. Pattern routing: use and theory for increasing predictability and avoiding coupling. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.21, n.7, p.777 –790, jul 2002.

KIRKPATRICK, S.; GELATT JR., C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, [S.l.], v.220, n.4598, p.671–680, May 1983.

LEE, T.-H.; CHANG, Y.-J.; WANG, T.-C. An enhanced global router with consideration of general layer directives. In: PHYSICAL DESIGN, 2011., New York, NY, USA. **Proceedings...** ACM, 2011. p.53–60. (ISPD '11).

LEE, Y.-T.; CHANG, Y.-J.; WANG, T.-C. A temperature-aware global router. In: VLSI DESIGN AUTOMATION AND TEST (VLSI-DAT), 2010 INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2010. p.279 –282.

LI, J.-T.; MAREK-SADOWSKA, M. Global Routing for Gate Array. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.3, n.4, p.298 – 307, oct 1984.

LILLIS, J. et al. New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing. In: DESIGN AUTOMATION CONFERENCE PROCEEDINGS 1996, 33RD. **Anais...** [S.l.: s.n.], 1996. p.395 –400.

LIU, W.-H. et al. Multi-threaded collision-aware global routing with bounded-length maze routing. In: DESIGN AUTOMATION CONFERENCE (DAC), 2010 47TH ACM/IEEE. **Anais...** [S.l.: s.n.], 2010. p.200 –205.

LUK, W. et al. A Hierarchical Global Wiring Algorithm for Custom Chip Design. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.6, n.4, p.518 – 533, july 1987.

MA, J.; HE, L. Towards global routing with RLC crosstalk constraints. In: DESIGN AUTOMATION CONFERENCE, 2002. PROCEEDINGS. 39TH. **Anais...** [S.l.: s.n.], 2002. p.669 –672.

MAREK-SADOWSKA, M. Route Planner for Custom Chip Design. In: IN INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN. **Anais...** IEEE, 1986. p.246–249.

MCMURCHIE, L.; EBELING, C. PathFinder: a negotiation-based performance-driven router for fpgas. In: FIELD-PROGRAMMABLE GATE ARRAYS, 1995. FPGA '95. PROCEEDINGS OF THE THIRD INTERNATIONAL ACM SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 1995. p.111 – 117.

MEIXNER, G.; LAUTHER, U. A new global router based on a flow model and linear assignment. In: COMPUTER-AIDED DESIGN, 1990. ICCAD-90. DIGEST OF TECHNICAL PAPERS., 1990 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1990. p.44 –47.

MOFFITT, M. MaizeRouter: engineering an effective global router. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.27, n.11, p.2017 –2026, nov. 2008.

MOFFITT, M. MaizeRouter: engineering an effective global router. In: DESIGN AUTOMATION CONFERENCE, 2008. ASPDAC 2008. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2008. p.226 –231.

OZDAL, M.; WONG, M. Archer: a history-driven global routing algorithm. In: COMPUTER-AIDED DESIGN, 2007. ICCAD 2007. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.488 –495.

OZDAL, M.; WONG, M. Archer: a history-based global routing algorithm. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.28, n.4, p.528 –540, april 2009.

PAN, M.; CHU, C. FastRoute: a step to integrate global routing into placement. In: COMPUTER-AIDED DESIGN, 2006. ICCAD '06. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. p.464 –471.

PAN, M.; CHU, C. FastRoute 2.0: a high-quality and efficient global router. In: DESIGN AUTOMATION CONFERENCE, 2007. ASP-DAC '07. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2007. p.250 –255.

POSSER, G. et al. A study on layout quality of automatic generated cells. In: ELECTRONICS, CIRCUITS, AND SYSTEMS (ICECS), 2010 17TH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.651 –654.

REIMANN, T.; SANTOS, G.; REIS, R. Routing algorithms performance in different routing scopes. In: ELECTRONICS, CIRCUITS, AND SYSTEMS (ICECS), 2010 17TH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.643–646.

REIS, R. Design automation of transistor networks, a new challenge. In: ISCAS. **Anais...** [S.l.: s.n.], 2011. p.2485–2488.

RICHARDS, D. Complexity of Single-Layer Routing. **Computers, IEEE Transactions on**, [S.l.], v.C-33, n.3, p.286 –288, Mar. 1984.

ROY, J.; MARKOV, I. High-performance routing at the nanometer scale. In: COMPUTER-AIDED DESIGN, 2007. ICCAD 2007. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.496 –502.

ROY, J.; MARKOV, I. High-Performance Routing at the Nanometer Scale. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.27, n.6, p.1066 –1077, june 2008.

SECHEN, C.; SANGIOVANNI-VINCENTELLI, A. TimberWolf3.2: a new standard cell placement and global routing package. In: DESIGN AUTOMATION, 1986. 23RD CONFERENCE ON. **Anais...** [S.l.: s.n.], 1986. p.432 – 439.

SHERWANI, N. A. **Algorithms for VLSI Physcial Design Automation**. 3rd.ed. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

SZYMANSKI, T. Dogleg Channel Routing is NP-Complete. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.4, n.1, p.31 – 41, january 1985.

TING, B.; TIEN, B. N. Routing Techniques for Gate Array. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.2, n.4, p.301 – 312, oct 1983.

VECCHI, M. P.; KIRKPATRICK, S. Global Wiring by Simulated Annealing. **IEEE Trans. on CAD of Integrated Circuits and Systems**, [S.l.], v.2, n.4, p.215–222, 1983.

VITTAL, A.; MAREK-SADOWSKA, M. Minimal Delay Interconnect Design Using Alphabetic Trees. In: DESIGN AUTOMATION, 1994. 31ST CONFERENCE ON. **Anais...** [S.l.: s.n.], 1994. p.392 – 396.

WANG, D.; KUH, E. Performance-driven interconnect global routing. In: VLSI, 1996. PROCEEDINGS., SIXTH GREAT LAKES SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 1996. p.132 –136.

WARME, D. M.; WINTER, P.; ZACHARIASEN, M. **GeoSteiner 3.1**. ., Department of Computer Science, University of Copenhagen (DIKU), Disponível em: `http://www.diku.dk/geosteiner/`. Acesso em: 25/02/2009.

WU, T.-H.; DAVOODI, A.; LINDEROTH, J. GRIP: scalable 3d global routing using integer programming. In: DESIGN AUTOMATION CONFERENCE, 2009. DAC '09. 46TH ACM/IEEE. **Anais...** [S.l.: s.n.], 2009. p.320 –325.

WU, T.-H.; DAVOODI, A.; LINDEROTH, J. GRIP: global routing via integer programming. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.30, n.1, p.72 –84, Jan. 2011.

WU, T.-H.; DAVOODI, A.; LINDEROTH, J. T. A parallel integer programming approach to global routing. In: DESIGN AUTOMATION CONFERENCE (DAC), 2010 47TH ACM/IEEE. **Anais...** [S.l.: s.n.], 2010. p.194 –199.

XIONG, J.; HE, L. Extended global routing with RLC crosstalk constraints. **Very Large Scale Integration (VLSI) Systems, IEEE Transactions on**, [S.l.], v.13, n.3, p.319 – 329, mar 2005.

XU, J. et al. An efficient hierarchical timing-driven Steiner tree algorithm for global routing. In: DESIGN AUTOMATION CONFERENCE, 2002. PROCEEDINGS OF ASP-DAC 2002. 7TH ASIA AND SOUTH PACIFIC AND THE 15TH INTERNATIONAL CONFERENCE ON VLSI DESIGN. PROCEEDINGS. **Anais...** [S.l.: s.n.], 2002. p.473 –478.

XU, J. et al. A novel timing-driven global routing algorithm considering coupling effects for high performance circuit design. In: DESIGN AUTOMATION CONFERENCE, 2003. PROCEEDINGS OF THE ASP-DAC 2003. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2003. p.847 – 850.

XU, J. et al. A coupling and crosstalk considered timing-driven global routing algorithm for high performance circuit design. In: DESIGN AUTOMATION CONFERENCE, 2004. PROCEEDINGS OF THE ASP-DAC 2004. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2004. p.677 – 682.

XU, J.; HONG, X.; JING, T. Timing-driven global routing with efficient buffer insertion. In: CIRCUITS AND SYSTEMS, 2005. ISCAS 2005. IEEE INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2005. p.2449 – 2452 Vol. 3.

XU, Y.; ZHANG, Y.; CHU, C. FastRoute 4.0: global router with efficient via minimization. In: DESIGN AUTOMATION CONFERENCE, 2009. ASP-DAC 2009. ASIA AND SOUTH PACIFIC. **Anais...** [S.l.: s.n.], 2009. p.576 –581.

YILDIZ, M.; MADDEN, P. Preferred direction Steiner trees. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.21, n.11, p.1368 – 1372, nov 2002.

YOUSSEF, H.; SAIT, S. M. Timing-driven global routing for standard-cell VLSI design. **International Journal of Computer Systems Science and Engineering**, [S.l.], v.14, n.3, p.175–185, May 1999.

ZHANG, L. et al. Performance and RLC crosstalk driven global routing. In: CIRCUITS AND SYSTEMS, 2004. ISCAS '04. PROCEEDINGS OF THE 2004 INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2004. v.5, p.V–65 – V–68 Vol.5.

ZHANG, Y.; XU, Y.; CHU, C. FastRoute3.0: a fast and high quality global router based on virtual capacity. In: COMPUTER-AIDED DESIGN, 2008. ICCAD 2008. IEEE/ACM INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.344 –349.

ZHOU, H. Efficient Steiner tree construction based on spanning graphs. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.23, n.5, p.704 – 710, may 2004.

ZHOU, H.; WONG, D. Global routing with crosstalk constraints. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.18, n.11, p.1683 –1688, nov 1999.

ZIESEMER, A.; LAZZARI, C. Transistor level automatic layout generator for non-complementary CMOS cells. In: VLSI-SOC. **Anais...** [S.l.: s.n.], 2007. p.116–121.