

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

**MÉTODOS HEURÍSTICOS APLICADOS AO PROBLEMA DA
ÁRVORE DE STEINER RECTILINEAR**

THIAGO GOUVEIA DA SILVA

João Pessoa-PB
Agosto-2009

THIAGO GOUVEIA DA SILVA

**MÉTODOS HEURÍSTICOS APLICADOS AO PROBLEMA DA
ÁRVORE DE STEINER RECTILINEAR**

Dissertação de Mestrado apresentada ao Centro de Ciências Exatas e da Natureza da Universidade Federal da Paraíba, como requisito parcial para obtenção do Título de Mestre em Informática (Sistemas de Computação).

Orientador: Prof. Dr. Lucídio dos Anjos Formiga Cabral

Banca Examinadora:

Prof. Dr. Lucídio dos Anjos Formiga Cabral (UFPB)

Prof. Dr. Antonio Carlos Cavalcanti (UFPB)

Prof. Dr. Júlio Francisco Barros Neto

João Pessoa-PB

Agosto-2009

S586m Silva, Thiago Gouveia da.
Métodos heurísticos aplicados ao problema da árvore de
Steiner rectilinear / Thiago Gouveia da Silva.- João Pessoa,
2009.
108f. : il.

Orientador: Lucídio dos Anjos Formiga Cabral
Dissertação (Mestrado) – UFPB/CCEN
1. Informática. 2. Árvores Retilíneas de Steiner. 3.
Metaheurística. 4. Simulated Annealing. 5. Algoritmos
Genéticos.

UFPB/BC

CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado do aluno Thiago Gouveia da Silva, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 28 de agosto de 2009.




1 Aos vinte e oito dias do mês de agosto do ano dois mil e nove, às nove horas, na Sala de
2 Reuniões do Centro de Ciências Exatas e da Natureza da Universidade Federal da Paraíba,
3 reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao
4 grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa
5 "Computação Distribuída", o Sr. Thiago Gouveia da Silva. A comissão examinadora foi
6 composta pelos professores doutores: Lucídio dos Anjos Formiga Cabral (DI-UFPB),
7 Orientador e Presidente da Banca Examinadora, Antonio Carlos Cavalcanti (DI-UFPB),
8 como examinador interno e Júlio Francisco Barros Neto (UFC), como examinador externo.
9 Dando início aos trabalhos, o Prof. Lucídio dos Anjos Formiga Cabral, cumprimentou os
10 presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato
11 para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado
12 "MÉTODOS HEURÍSTICOS APLICADOS AO PROBLEMA DA ÁRVORE DE STEINER
13 RECTILINEAR". Concluída a exposição, o candidato foi argüido pela Banca Examinadora
14 que emitiu o seguinte parecer: "aprovado". Assim sendo, deve a Universidade Federal da
15 Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para
16 constar, eu, professora Tatiana Aires Tavares, coordenadora do Programa de Pós-
17 Graduação em Informática, servindo de secretária, lavrei a presente ata que vai assinada por
18 mim mesmo e pelos membros da Banca Examinadora. João Pessoa, 28 de agosto de 2009.

19 
20 Tatiana Aires Tavares

21
Prof. Dr. Lucídio dos Anjos Formiga Cabral
Orientador (DI-UFPB)

Prof. Dr. Antonio Carlos Cavalcanti
Examinador Interno (DI-UFPB)

Prof. Dr. Júlio Francisco Barros Neto
Examinador Externo (UFC)

RESUMO

RESUMO

Este trabalho apresenta uma nova heurística, denominada Heurística 1, e a implementação das metaheurísticas GRASP, *Simulated Annealing* e Algoritmos Genéticos para o problema da árvore retilínea mínima de Steiner (RSMTP), discutindo sobre seus aspectos teóricos, como a complexidade computacional; e práticos, como pseudocódigos e estratégias de implementação. As novas abordagens para o RSMTP apresentadas, em especial os Algoritmos Genéticos, ostentam resultados computacionais de qualidade superior às apresentadas pelas melhores heurísticas da literatura atual.

Palavras-chave: Árvores Retilíneas de Steiner, Metaheurística, *Simulated Annealing*, GRASP, Algoritmos Genéticos.

ABSTRACT

This work presents a new heuristic, called Heurística 1, and the implementations of the GRASP, Simulated Annealing and Genetic Algorithms metaheuristics for the rectilinear Steiner minimum tree problem (RSMTP), talking about its theoretical aspects, like computational complexity, and practical ones, like pseudo-codes and implementation strategies. The new techniques for RSMTP presented, especially the Genetic Algorithms, have computational results of superior quality in comparison to the best heuristics in present literature.

Key words: Rectilinear Steiner Trees, Metaheuristic, Simulated Annealing, GRASP, Genetic Algorithms.

AGRADECIMENTOS

Agradeço

Primeiramente a DEUS, por ter me feito companhia em cada madrugada fria de trabalho.

A meu pai – Antônio Gabriel, minha mãe – Josélia, minha irmã – Gaby e minha avó – Hilda, pelo apoio incondicional em cada etapa da construção deste trabalho.

Ao Prof. Dr. Lucídio, pelo laço de amizade construído durante estes anos e por acreditar, sempre, na conclusão deste trabalho.

Ao Professor Hélio, por sempre demonstrar confiança na minha capacidade.

Aos meus amigos Nailson, Pedoca, Curuma, Alysson, Alan e Niltinho, por compreenderem este longo período de ausências.

Aos amigos da CAGEPA: Helton, Márcio, Leonardo, Eduardo e Renatão.

Aos amigos da SEFAZ-PE: Rafael, Amaro, Paulo, Ricardo e Marcelo Rosas.

As empresas ATI-PE e SEFAZ-PE, pela dispensa de oito horas semanais para conclusão deste trabalho.

Especialmente a Thiago Curvelo, pela amizade, paciência e dedicação no apoio a este que vos escreve, não só em termos de ciência, mas em todos os âmbitos da vida.

SUMÁRIO

RESUMO.....	IV
AGRADECIMENTOS	V
SUMÁRIO	VI
LISTA DE FIGURAS.....	IX
LISTA DE TABELAS	XI
LISTA DE ACRÔNIMOS.....	XII
CAPÍTULO 1 - INTRODUÇÃO.....	1
1.1 DELIMITAÇÃO DE OBJETO	3
1.1.1 <i>Objetivos Gerais</i>	3
1.1.2 <i>Objetivos Específicos</i>	3
1.2 METODOLOGIA	4
1.2.1 <i>Atividades</i>	4
1.2.2 <i>Ambiente de Desenvolvimento</i>	5
CAPÍTULO 2 - FUNDAMENTAÇÃO TEÓRICA	7
2.1 FUNDAMENTOS SOBRE HEURÍSTICAS E METAHEURÍSTICAS	7
2.2 A METAHEURÍSTICA GRASP	9
2.3 A METAHEURÍSTICA SIMULATED ANNEALING.....	12
2.4 ALGORITMOS GENÉTICOS	16
2.4.1 <i>Representação de Indivíduos</i>	18
2.4.2 <i>Processo de Geração da População Inicial</i>	19
2.4.3 <i>Etapa de Reprodução</i>	20
2.4.3.1 Operador Clássico de Recombinação.....	21
2.4.3.2 Operador Clássico de Mutação	21
2.4.4 <i>Critérios de Parada</i>	22
2.4.5 <i>Problemas de Convergência</i>	22
2.5 FUNDAMENTOS SOBRE ÁRVORE RETILÍNEA MÍNIMA DE STEINER.....	23
2.5.1 <i>Teorema da grade de Hanan</i>	24

2.5.2 Tipos de pontos de Steiner	25
2.5.3 Topologias de árvores de Steiner	26
2.5.4 Árvores Retilíneas Full-Steiner	27
2.5.5 Algoritmos de Árvore Geradora Retilínea Mínima	28
2.5.6 Half Perimeter Wirelength	30
2.5.7 Aresta e Distância Gargalo.....	31
2.5.8 Redução de Pontos.....	31
CAPÍTULO 3 - O ESTADO DA ARTE	33
3.1 BIIS	34
3.1.1 Independência entre pontos.....	36
3.1.2 Manutenção dinâmica da RMST	37
3.1.3 Outras melhorias no BIIS.....	38
3.2 BGA.....	38
3.3 FLUTE.....	42
3.3.1 Sequência Vertical	42
3.3.2 Distância Marginal.....	43
3.3.3 Vetores de Coeficientes Potencialmente Ótimos	44
3.3.4 FLUTE para Instâncias com Dez ou Mais Terminais	45
3.3.5 Complexidade Computacional e Resultados	47
3.4 GEOSTEINER.....	48
3.4.1 Algoritmo de Geração de RFST	48
3.4.2 Concatenação de RFSTs	49
3.4.3 Outras Utilidades do conjunto de RFSTs	49
CAPÍTULO 4 - MÉTODOS PROPOSTOS	51
4.1 NOTAÇÕES	51
4.2 HEURÍSTICA1 – H1	52
4.2.1 Procedimento de Geração de RMST	53
4.2.2 Localização de Pontos de Steiner Candidatos	54
4.2.3 Complexidade Computacional	55
4.2.4 Exemplo Prático.....	55

4.3 GRASP	56
4.3.1 <i>H1 Multi-Start</i>	56
4.3.2 <i>GRASP-H1</i>	58
4.3.3 <i>Complexidade Computacional</i>	60
4.4 SIMULATED ANNEALING	61
4.4.1 <i>Movimento Edge_Swap</i>	64
4.4.2 <i>Movimento Add_Steiner_3</i>	65
4.4.3 <i>Movimento Del_Steiner</i>	66
4.4.4 <i>Movimento Graceful_Del_Steiner</i>	68
4.4.5 <i>Movimento Pierce_Steiner</i>	68
4.4.6 <i>Complexidade Computacional</i>	70
4.5 ALGORITMO GENÉTICO	70
4.5.1 <i>Representação Genética das Soluções</i>	71
4.5.2 <i>Função de Aptidão de um Indivíduo</i>	74
4.5.3 <i>Procedimentos de Geração Populacional</i>	74
4.5.3.1 <i>Procedimento de Recombinação Odd_Even_Crossover</i>	74
4.5.3.2 <i>Procedimento de Mutação hy-M</i>	75
4.5.3.3 <i>Ilustração da Etapa de Reprodução do GA</i>	76
4.5.4 <i>Definição da População Sobrevivente</i>	76
CAPÍTULO 5 - RESULTADOS COMPUTACIONAIS	78
5.1 <i>SIMULAÇÃO COM INSTÂNCIAS GERADAS ALEATORIAMENTE</i>	78
5.2 <i>SIMULAÇÃO COM INSTÂNCIAS BENCHMARK DA OR_LIB</i>	81
CAPÍTULO 6 - CONSIDERAÇÕES FINAIS E PROPOSTA DE TRABALHOS	
FUTUROS	89
REFERÊNCIAS.....	91

LISTA DE FIGURAS

FIGURA 2.1: EXEMPLO DE VARIAÇÃO DO CUSTO PELO ESPAÇO DE SOLUÇÕES.....	9
FIGURA 2.2: PSEUDOCÓDIGO DA METAHEURÍSTICA GRASP	10
FIGURA 2.3: PROCEDIMENTO CONSTRUTIVO GRASP	11
FIGURA 2.4: COMPORTAMENTO DO FATOR DE BOLTZMANN EM RELAÇÃO À TEMPERATURA..	13
FIGURA 2.5: PSEUDOCÓDIGO DA METAHEURÍSTICA <i>SIMULATED ANNEALING</i>	15
FIGURA 2.6: PSEUDOCÓDIGO DOS ALGORITMOS GENÉTICOS	18
FIGURA 2.7: GERAÇÃO UNIFORME DA POPULAÇÃO INICIAL.....	19
FIGURA 2.8: GERAÇÃO DA POPULAÇÃO INICIAL POR INVERSÃO	20
FIGURA 2.9: (A) ÁRVORE EUCLIDIANA, (B) ÁRVORE RETILÍNEA E (C) ÁRVORE RETILÍNEA DE STEINER.....	23
FIGURA 2.10: TEOREMA DA GRADE DE HANAN	25
FIGURA 2.11: EXEMPLOS DE PONTOS DE STEINER.....	25
FIGURA 2.12: REMOÇÃO DE UM <i>CORNER-POINT</i>	26
FIGURA 2.13: DIFERENTES TOPOLOGIAS DE UMA MESMA ÁRVORE RETILÍNEA DE STEINER	27
FIGURA 2.14: TOPOLOGIAS DE HWANG PARA RFSTs	27
FIGURA 2.15: OCTANTES DE UM PONTO QUALQUER.....	29
FIGURA 2.16: TRÊS FASES DA REDUÇÃO DE PONTOS E ARESTAS DE WINTER	32
FIGURA 2.17: REGIÕES VAZIAS: A) DIAMANTE, B) RETÂNGULO, C) TRIÂNGULO E D) CÍRCULO	32
FIGURA 3.1: CLASSES DE TÉCNICAS PARA O RSMT	33
FIGURA 3.2: EXECUÇÃO SIMPLES DO IIS.....	35
FIGURA 3.3: QUADRANTES DEFINIDOS PELA PARTIÇÃO DIAGONAL DE UM PONTO	37
FIGURA 3.4: MANUTENÇÃO DINÂMICA DA RMST	38
FIGURA 3.5: ÁRVORE 3-RESTRITA	39
FIGURA 3.6: PSEUDOCÓDIGO DO HGP [<i>KAHNG, MANDOIU E ZELIKOVSKY, 2003</i>]	41
FIGURA 3.7: ROTINA DE COMPUTAÇÃO DA ARESTA DE GARGALO DO CAMINHO ENTRE DOIS TERMINAIS U E V	41
FIGURA 3.8: SEQUÊNCIA VERTICAL	43
FIGURA 3.9: DISTÂNCIA MARGINAL	44
FIGURA 3.10: VETORES DE COEFICIENTES.....	44

FIGURA 3.11: QUEBRA DE INSTÂNCIA	46
FIGURA 3.12: AUMENTANDO UMA RFST	48
FIGURA 4.1: PSEUDOCÓDIGO DA HEURÍSTICA 1	52
FIGURA 4.2: PROCEDIMENTO DE GERAÇÃO DE RMST	53
FIGURA 4.3: GRADE DE HANAN k -RESTRITA	54
FIGURA 4.4: EXEMPLO DE EXECUÇÃO DA H1	56
FIGURA 4.5: DIFERENTES RMSTs PARA UM MESMO CONJUNTO DE TERMINAIS	57
FIGURA 4.6: PROCEDIMENTO H1 <i>MULTI-START</i>	58
FIGURA 4.7: PROCEDIMENTO GRASP DE GERAÇÃO DE ÁRVORES RETILÍNEAS	59
FIGURA 4.8: COMPARAÇÃO ENTRE OS PROCEDIMENTOS H1 <i>MULTI-START</i> E GRASP-H1	60
FIGURA 4.9: METAHEURÍSTICA HÍBRIDA SA / <i>MULTI-START</i>	61
FIGURA 4.10: PSEUDOCÓDIGO DO <i>SIMULATED ANNEALING</i>	62
FIGURA 4.11: PSEUDOCÓDIGO DO MOVIMENTO <i>EDGE_SWAP</i>	64
FIGURA 4.12: EXEMPLO DE EXECUÇÃO DO MOVIMENTO <i>EDGE-SWAP</i>	65
FIGURA 4.13: PSEUDOCÓDIGO DO MOVIMENTO <i>ADD_STEINER_3</i>	65
FIGURA 4.14: EXEMPLO DE EXECUÇÃO DO MOVIMENTO <i>ADD_STEINER_3</i>	66
FIGURA 4.15: PSEUDOCÓDIGO DO MOVIMENTO <i>DEL_STEINER</i>	67
FIGURA 4.16: EXEMPLO DE EXECUÇÃO DO MOVIMENTO <i>DEL_STEINER</i>	68
FIGURA 4.17: EXEMPLO DE EXECUÇÃO DO MOVIMENTO <i>PIERCE-STEINER</i>	69
FIGURA 4.18: IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO	71
FIGURA 4.19: REPRESENTAÇÃO GENÉTICA DE ÁRVORES DE STEINER	72
FIGURA 4.20: REPRESENTAÇÃO DE UMA POPULAÇÃO DE ÁRVORES DE STEINER	73
FIGURA 4.21: PROCEDIMENTO DE RECOMBINAÇÃO <i>ODD_EVEN_CROSSOVER</i>	75
FIGURA 4.22: EXECUÇÃO DO PROCEDIMENTO DE REPRODUÇÃO	76
FIGURA 5.1: RELAÇÃO ENTRE O GAP PARA O ÓTIMO E O NÚMERO DE PONTOS DAS INSTÂNCIAS	83
FIGURA 5.2: MELHORIA MÉDIA SOBRE A RMST	84

LISTA DE TABELAS

TABELA 2.1: TEMPO DE EXECUÇÃO MÉDIO DOS ALGORITMOS DE GERAÇÃO DE RMST	30
TABELA 3.1: NÚMERO DE ITERAÇÕES E NÚMERO DE PONTOS ADICIONADOS POR ITERAÇÃO NO BIIS	36
TABELA 3.2: RELAÇÃO DE POWVs POR GRUPO	45
TABELA 3.3: PARÂMETRO A	47
TABELA 5.1: DETALHAMENTO DO GRUPO DE INSTÂNCIAS <i>RAND_LIB</i>	79
TABELA 5.2: <i>GAP</i> PARA O ÓTIMO DAS TÉCNICAS H1, GRASP, SA E GA SOBRE A <i>RAND_LIB</i> .	79
TABELA 5.3: <i>GAP</i> PARA A RMST DAS TÉCNICAS H1, GRASP, SA E GA SOBRE A <i>RAND_LIB</i>	80
TABELA 5.4: COMPARATIVO DO <i>GAP</i> PARA O ÓTIMO ENTRE O GA E O SA	81
TABELA 5.5: : DETALHAMENTO DO GRUPO DE INSTÂNCIAS <i>BENCHMARK</i> DA <i>OR_LIB</i>	82
TABELA 5.6: TABELA-RESUMO DO <i>GAP</i> PARA O ÓTIMO	83
TABELA 5.7: TABELA-RESUMO DO <i>GAP</i> SOBRE A RMST	84
TABELA 5.8: RESULTADOS OBTIDOS PARA O GRUPO 100 DA <i>OR_LIB</i>	85
TABELA 5.9: RESULTADOS OBTIDOS PARA O GRUPO 250 DA <i>OR_LIB</i>	86
TABELA 5.10: RESULTADOS OBTIDOS PARA O GRUPO 500 DA <i>OR_LIB</i>	87
TABELA 5.11: RESULTADOS OBTIDOS PARA O GRUPO 1000 DA <i>OR_LIB</i>	88

LISTA DE ACRÔNIMOS

BIIS	BATCHED ITERATED 1-STEINER
BGA	BATCHED GREEDY ALGORITHM
FLUTE	FAST LOOKUP TABLE BASED WIRELENGTH ESTIMATION TECHNIQUE
FLUTE-MR	FLUTE MERGE REDUNDANT
FLUTE-AM	FLUTE AGGRESSIVE MERGE
FPGA	FIELD PROGRAMMABLE GATE ARRAY
GA	GENETIC ALGORITHM
GTCA	GREEDY TRIPLE CONTRACTION ALGORITHM
GRASP	GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURES
H1	HEURÍSTICA 1
HGP	HIERARQUICAL GREEDY PREPROCESSING
HPWL	HALF PERIMETER WIRELENGTH
IIS	ITERATED 1-STEINER
MST	MINIMUM SPANNING TREE
MSTP	MINIMUM SPANNING TREE PROBLEM
POWV	POTENTIALLY OPTIMAL WIRELENGTH VECTOR
RFST	RECTILINEAR FULL STEINER TREE
RMST	RECTILINEAR MINIMUM SPANNING TREE
RSMT	RECTILINEAR STEINER MINIMUM TREE PROBLEM
SA	SIMULATED ANNEALING
SMT	STEINER MINIMUM TREE PROBLEM
VLSI	VERY-LARGE-SCALE INTEGRATION
VNS	VARIABLE NEIGHBORHOOD SEARCH

CAPÍTULO 1 - INTRODUÇÃO

O problema da árvore geradora mínima de Steiner (*Steiner Minimum Tree Problem* - SMTP), de modo similar ao problema da árvore geradora mínima (*Minimum Spanning Tree Problem* - MSTP), consiste em: dado um conjunto P de pontos no plano cartesiano, interconectá-los através da árvore A de menor comprimento. O comprimento de uma árvore é dado pela soma dos custos de todas as arestas desta. A diferença entre o SMTP e o MSTP é que, no SMTP, pontos extras podem ser adicionados a P com o intuito de diminuir o comprimento de A . Estes pontos extras são denominados pontos de Steiner, enquanto a árvore resultante é chamada árvore de Steiner.

Apesar de o MSTP possuir famosas soluções em tempo polinomial - os algoritmos de Kruskal e de Prim, ambos apresentando complexidade $O(n \log(n))$ [Cormen et al., 2001] - o SMTP é NP-difícil [Garey et al., 1977]. Esta classe de problemas caracteriza-se pela não existência de algoritmos de tempo polinomial, o que propicia espaço para a utilização de abordagens semi-exatas.

O objeto deste trabalho é o problema da árvore geradora mínima de Steiner em distância retilínea (*Rectilinear Steiner Minimum Tree Problem* - RSMTP), variante do SMTP na qual a distância entre os pontos é aferida mediante métrica L1, distância de Manhattan. O RSMTP possui várias aplicações no projeto de chips VLSI (*Very-large-scale integration*). Nas fases de síntese e posicionamento pode ser usado para estimar o comprimento final do circuito, o congestionamento e o atraso das interconexões; enquanto nas fases de roteamento global e detalhado, é usado para gerar a topologia de roteamento de cada rede.

Mesmo com as novas funções-objetivo para roteamento introduzidas pelos recentes avanços na tecnologia dos circuitos integrados (especialmente em escala *deep-submicron*), o RSMTP mantém sua relevância [Mandoiu et al., 1999]: para redes não-críticas, ou instâncias fisicamente pequenas, a minimização do

comprimento das interconexões significa a diminuição da capacitância e da área total do circuito.

Assim como o SMTP, o RSMTP é NP-difícil [Garey et al., 1977], e muito esforço tem sido dedicado ao desenvolvimento de algoritmos e heurísticas para este problema. Entre as abordagens semi-exatas, o *Batched Iterated First Steiner* (BI1S) [Kahng e Robins, 1992] ostentou os resultados mais precisos por muitos anos. Recentemente, porém, o BI1S foi superado pelo *Fast Lookup Table Based Wirelength Estimation Technique* (FLUTE) [Chu, 2004]. Entre as técnicas exatas, o GeoSteiner [Warme, Winter e Zachariasen, 1998] apresenta o menor tempo de execução médio para instâncias aleatórias.

Nesse contexto, combinaram-se alguns avanços anteriores a esta dissertação com abordagens heurísticas e metaheurísticas, concebendo um conjunto de novas estratégias para o RSMTP. Estas novas abordagens foram encapsuladas em um aplicativo denominado NeoSteiner.

Em um primeiro momento, desenvolveu-se uma heurística simples e rápida (denominada H1) que, assim como as primeiras heurísticas propostas para o RSMTP, é baseada no algoritmo da árvore geradora mínima de Prim.

Posteriormente, foram aplicadas as metaheurísticas GRASP (*Greedy Randomized Adaptive Search Procedures*), SA (*Simulated Annealing*) e GA (*Genetic Algorithms*) ao RSMTP. Diferentemente do SMTP, não se encontrou referência a abordagens metaheurísticas aplicadas ao RSMTP na literatura.

Os resultados obtidos pelo NeoSteiner mostraram-se bastante competitivos com o estado da arte, seja em tempo de execução (caso do H1 e do GRASP) ou em qualidade da solução (caso do SA e do GA).

Com o intuito de descrever da melhor maneira o trabalho realizado, dividiu-se esta dissertação em seis capítulos.

Esta introdução (primeiro capítulo) apresenta a delimitação do escopo e os objetivos gerais e específicos deste trabalho, assim como a metodologia utilizada para o seu desenvolvimento.

O segundo capítulo, responsável pela fundamentação teórica, clarifica os conceitos-chave necessários para a compreensão das estratégias aqui propostas.

O terceiro capítulo apresenta uma importante revisão das heurísticas (e do algoritmo GeoSteiner) que compõem o estado da arte para o RSMT. Algumas técnicas utilizadas pelo NeoSteiner, tais como a redução de pontos e a manutenção dinâmica da árvore geradora mínima (*Minimum Spanning Tree* - MST), são baseadas nestes trabalhos.

O quarto capítulo, espinha dorsal desta dissertação, apresenta o estudo, a implementação, o pseudocódigo e a análise da complexidade computacional da heurística H1 e das metaheurísticas GRASP, SA e GA aplicadas ao RSMT.

O quinto capítulo apresenta os resultados obtidos pelo NeoSteiner em comparação às abordagens descritas no capítulo três.

O sexto capítulo apresenta a conclusão, as considerações finais e as propostas de trabalhos futuros, sendo seguido pelas referências bibliográficas.

1.1 Delimitação de Objeto

1.1.1 Objetivos Gerais

Compreender o RSMT e suas aplicações, a fim de manter o foco no estudo e desenvolvimento de técnicas adequadas ao cenário atual.

Adquirir um conhecimento mais profundo nas áreas de inteligência computacional, com ênfase nas metaheurísticas SA, GRASP e GA.

Estudar, planejar e implementar novas técnicas para o RSMT, com o intuito de melhorar as abordagens existentes em custo computacional e/ou qualidade de solução.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho descrevem as metas a serem alcançadas, tendo em vista o objeto de estudo desta dissertação. Para tanto, estão previstos os seguintes pontos:

- Estudar as melhores heurísticas e algoritmos da literatura e identificar o que de melhor possuem.

- Desenvolver uma heurística rápida e robusta o suficiente para ser usada como estimativa de roteamento em chips VLSI.
- Implementar e parametrizar as metaheurísticas SA, GRASP e GA para o RSMTP.
- Realizar testes e simulações para comparar os resultados obtidos com o estado da arte, comprovando, assim, a eficácia da heurística desenvolvida e das metaheurísticas aplicadas.

1.2 Metodologia

A realização deste trabalho foi executada mediante uma metodologia “*top-down*”, na qual é buscada, a princípio, uma compreensão holística do problema, seguida pela prospecção das soluções existentes, para, só então, realizar-se o projeto e a implementação de uma nova solução.

1.2.1 Atividades

Para atingir os objetivos supracitados foram realizadas as seguintes atividades:

- Obtenção de subsídios teóricos sobre RSMTP, visando melhor compreensão dos seus pilares matemáticos.
- Levantamento bibliográfico direcionado à identificação das melhores soluções da Literatura.
- Obtenção e estudo das melhores soluções do RSMTP, tanto em código-fonte quanto em publicações nas quais são descritas.
- Estudo sobre algoritmos, heurísticas e metaheurísticas, objetivando a definição da estratégia mais eficiente de resolução do RSMTP.
- Definição, estudo, projeto e implementação da abordagem heurística H1.
- Definição, estudo, projeto, implementação e parametrização das metaheurísticas GRASP, SA e GA aplicadas ao RSMTP.

- Encapsulamento das soluções implementadas em um único aplicativo, denominado NeoSteiner.
- Utilização do aplicativo gprof [Fenlason e Stallman, 2008] para identificar quais métodos do NeoSteiner consomem mais tempo de processamento, facilitando, assim, a redução do tempo total de execução do aplicativo.
- Utilização do aplicativo valgrind [Seward et al., 2008] para identificar e remover usos indevidos de memória, eventualmente presentes no NeoSteiner.
- Realização de testes para geração de tabelas e gráficos comparativos entre o NeoSteiner e o estado da arte.
- Avaliação dos resultados obtidos, levantamento dos pontos fortes e fracos do NeoSteiner e proposição de trabalhos futuros.
- Publicação dos resultados obtidos.
- Construção do manual de utilização do NeoSteiner.
- Redação desta dissertação.

1.2.2 Ambiente de Desenvolvimento

O NeoSteiner foi escrito em linguagem de programação “C” e compilado utilizando o gcc versão “Debian 4.3.3-10”, em modo de otimização de código “O3”. O computador no qual se deu o processo de projeto e desenvolvimento, assim como de realização dos testes comparativos, apresenta as seguintes características:

- Sistema operacional Debian Linux (kernel 2.6.18);
- Processador Intel® Pentium® 4 CPU 2.80GHz;
- Memória cache L1 de 1024 KB;
- 2048 KB de memória Ram;

A visualização das soluções é propiciada pelo o aplicativo gnuplot, interligado ao NeoSteiner através da biblioteca gnuplot_i.

A execução de testes e a geração das tabelas e gráficos deste trabalho foram automatizadas através de scripts shell e awk.

CAPÍTULO 2 - FUNDAMENTAÇÃO TEÓRICA

Este capítulo destina-se a elucidar os conceitos centrais deste trabalho, oferecendo, assim, os subsídios teóricos necessários à sua completa compreensão.

A primeira seção traz a definição de heurística e metaheurística, tal como uma breve explicação das metaheurísticas GRASP, SA e GA.

A seção subsequente, além definir formalmente o RSMTP, apresenta alguns avanços matemáticos e geométricos utilizados para uma solução mais eficiente deste problema.

2.1 Fundamentos Sobre Heurísticas e Metaheurísticas

Muitos problemas práticos existentes no dia a dia podem ser solucionados otimamente por meio da computação, i.e., é possível encontrar a melhor solução existente para tal problema.

Há problemas, porém, que por apresentar características combinatórias, não possuem algoritmos capazes de encontrar a sua solução ótima em tempo polinomial. Estes problemas constituem uma classe denominada NP-difícil.

Em problemas dessa natureza, onde métodos exatos tornam-se inviáveis, pode-se encontrar, todavia, boas soluções em tempo razoável. Em geral, tais soluções apresentam uma relação custo/benefício bastante vantajosa, visto que o esforço computacional exigido para encontrar a solução ótima pode demandar, em muitos casos, anos, décadas ou séculos de processamento (tempos impraticáveis computacionalmente).

O conceito de heurística é definido como uma técnica inspirada em processos intuitivos que procura uma solução de boa qualidade a um custo computacional aceitável, sem, no entanto, estar capacitada a garantir a solução ótima, bem como garantir quão próximo está desta [Souza, 2008].

O desafio é produzir, em tempo reduzido, soluções tão próximas quanto possível do ótimo. Muitos esforços têm sido feitos nesta direção e heurísticas muito eficientes foram desenvolvidas para diversos problemas. Entretanto, a maioria das heurísticas desenvolvidas é específica para um problema particular, não sendo eficiente, ou mesmo aplicável, à resolução de uma classe mais ampla de problemas.

As heurísticas são classificadas em duas classes: heurísticas construtivas e heurísticas de refinamento.

As heurísticas construtivas, como o nome sugere, são responsáveis pela construção, elemento por elemento, de uma solução. A seleção do elemento a ser inserido em cada passo, varia de acordo com a função de avaliação adotada, que, por sua vez, depende do problema abordado. Nas heurísticas clássicas, os elementos candidatos são geralmente ordenados segundo uma função gulosa, que estima o benefício da inserção de cada elemento, e somente o “melhor” elemento é inserido a cada passo desta.

As heurísticas de refinamento -- também conhecidas como técnicas de busca local -- partem de uma solução inicial qualquer, obtida através de uma heurística construtiva ou gerada aleatoriamente, e caminham iterativamente sobre a vizinhança desta solução a fim de encontrar melhores resultados.

[Souza, 2008] define o conceito de vizinhança como segue: seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a minimizar, a função N , a qual depende da estrutura do problema tratado, associa a cada solução $s \in S$, sua vizinhança $N(s) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s . Denomina-se movimento a modificação que transforma uma solução $s \in S$ em outra, $s' \in N(s)$.

A partir da década de 80 surgiu uma nova classe de heurísticas, reunindo conceitos das áreas de otimização e inteligência artificial, denominada metaheurística. Caracteriza-se como um conjunto de conceitos usados para definir métodos heurísticos que podem ser aplicados em vários problemas diferentes de otimização.

Uma característica comum às metaheurísticas é a capacidade de escapar de ótimos locais a fim de continuar a busca pelo ótimo global. Tal atributo não é compartilhado com os métodos heurísticos. A Figura 2.1 exemplifica a presença de diversos mínimos locais em um dado espaço de soluções hipotético.

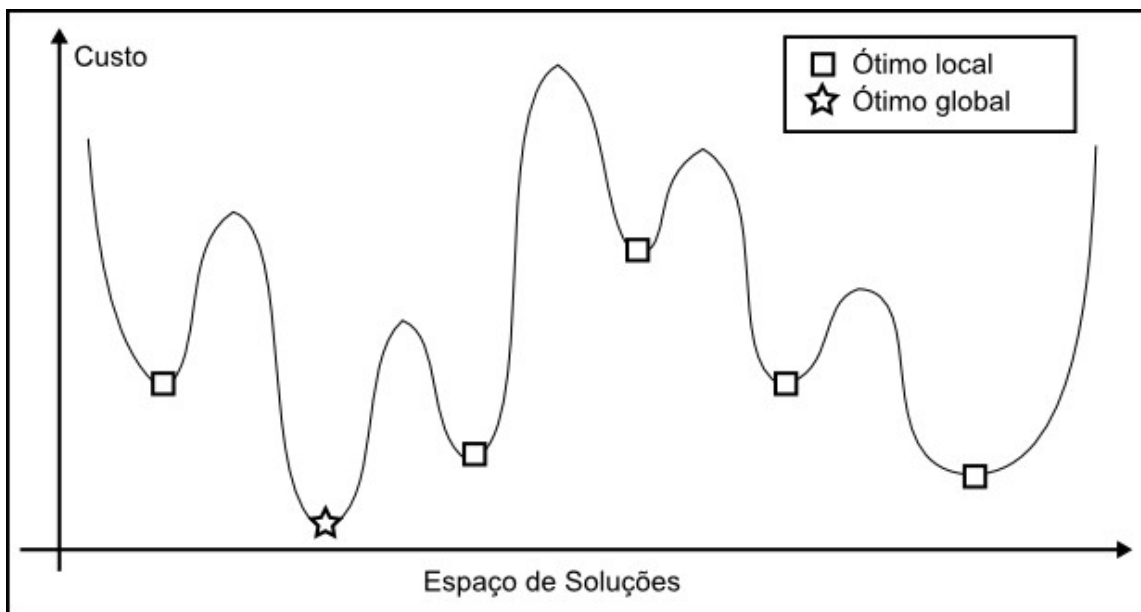


Figura 2.1: Exemplo de variação do custo pelo espaço de soluções

Define-se: metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico [Ribeiro, 1996].

Dentre os procedimentos enquadrados como metaheurísticas destacam-se: algoritmos genéticos, redes neurais, *Simulated Annealing*, busca tabu, GRASP, VNS (*Variable Neighborhood Search*) e colônia de formigas.

2.2 A Metaheurística GRASP

O GRASP (*Greedy Randomized Adaptive Search Procedure*), em português “Procedimento de Busca Gulosa Adaptativa Aleatória”, é uma metaheurística *multi-start* iterativa na qual cada iteração é composta por uma fase de construção e uma fase de busca local [Resende, 2003].

O GRASP foi proposto inicialmente por [Feo et al., 1994] e, atualmente, possui um grande destaque na literatura devido aos bons resultados obtidos em problemas de otimização [Rocha e Alvarenga, 2006]. A Figura 2.2 mostra o pseudocódigo do procedimento GRASP.

Procedimento GRASP

```
1 ENQUANTO (condição de parada não for satisfeita), FAÇA
2   solução = crie uma solução construtiva_ateatoria();
3   solução = busca_local(solução);
4   SE solução é a melhor solução até então conhecida ENTÃO
5     grave(solução);
6   FIM SE
7 FIM ENQUANTO
```

Figura 2.2: Pseudocódigo da metaheurística GRASP

No GRASP, a fase de construção é iterativa, adaptativa, semi-gulosa e randômica, gerando soluções viáveis para o problema através de um procedimento parcialmente guloso e parcialmente aleatório [Ferreira e Ochi, 2007].

A cada etapa da construção da solução, seleciona-se uma lista restrita, formada pelos $a\%$ melhores candidatos, baseada num critério de ordenação pré-definido. Sobre esta lista é feita aleatoriamente a escolha do próximo elemento a compor a solução.

O parâmetro a (no intervalo $[0,1]$) controla o nível de “gulosidade” e aleatoriedade do procedimento de construção. Um valor $a = 0$ faz gerar soluções puramente gulosas, enquanto $a = 1$ faz produzir soluções totalmente aleatórias.

A metaheurística GRASP é dita adaptativa pois os benefícios associados com a escolha de cada elemento são atualizados em cada iteração da fase de construção para refletir as mudanças oriundas da seleção do elemento anterior [Souza, 2008].

A aleatoriedade da escolha dos elementos permite a visita de uma porção maior do universo de soluções. Essa característica permite que o procedimento

GRASP escape de ótimos locais. O procedimento de construção da solução é ilustrado na Figura 2.3.

A fase de refinamento torna-se bastante importante devido às características aleatórias da fase de construção, que, assim como muitos procedimentos não determinísticos, não nos encaminham para soluções localmente ótimas.

Nesta fase podem ser utilizadas quaisquer heurísticas de busca local, que tem como objetivo melhorar uma solução inicial até que ela atinja um ótimo local.

Procedimento Construtivo GRASP

```
1 Inicialize o conjunto C de Candidatos;  
2 ENQUANTO C diferente de vazio FAÇA  
3   Ordene a lista de Candidatos;  
4   Monte a Lista_Restrita com os ALPHA melhores Candidatos;  
5   Selecione aleatoriamente um membro da Lista_Restrita, K;  
6   Adicione K à solução;  
7   Atualize o conjunto C;  
8 FIM ENQUANTO  
9 Retorne a solução obtida;
```

Figura 2.3: Procedimento construtivo GRASP

O parâmetro α é basicamente o único parâmetro a ser ajustado na implementação de um procedimento GRASP. Valores de α que levam a uma lista de candidatos restrita de tamanho muito limitado implicam em soluções finais de qualidade muito próxima àquela obtida de forma puramente gulosa, obtidas com um baixo esforço computacional. Em contrapartida, provocam uma baixa diversidade de soluções construídas. Já uma escolha de α próxima da seleção puramente aleatória leva a uma grande diversidade de soluções construídas, por outro lado, muitas das soluções construídas são de qualidade inferior, tornando mais lento o processo de busca local.

O procedimento GRASP bem parametrizado e implementado agrega os bons aspectos dos algoritmos gulosos e dos algoritmos de construção aleatória, constituindo-se uma metaheurística rápida e poderosa.

As principais características do GRASP são:

- Facilidade de implementação;
- Facilidade de parametrização, devido ao único parâmetro α ;
- Facilidade de implementação paralela, devido à relativa independência entre suas iterações.

2.3 A Metaheurística Simulated Annealing

A metaheurística SA (*Simulated Annealing*), proposta originalmente por [Kirkpatrick et al., 1983], fundamenta-se em uma analogia com a termodinâmica, ao simular o arrefecimento de um conjunto de átomos aquecidos.

O nome recozimento (*annealing*) é dado ao processo de aquecimento de um sólido até o seu ponto de fusão, seguido de um resfriamento gradual e vagaroso, até que se alcance novamente o seu enrijecimento. Nesse processo, o lento resfriamento é essencial para se manter um equilíbrio térmico onde os átomos encontrarão tempo suficiente para se organizarem em uma estrutura uniforme e com energia mínima. Se o sólido é resfriado bruscamente, seus átomos formarão uma estrutura irregular e fraca [Mazzucco Junior, 1999].

O recozimento pode ser visto como um processo estocástico de determinação de uma organização dos átomos de um sólido que apresente energia mínima.

Em temperaturas altas, os átomos se movem livremente e, com grande probabilidade, podem mover-se para posições que incrementarão a energia total do sistema.

Quando se tem baixas temperaturas, os átomos gradualmente se movem em direção a uma estrutura regular e, somente com pequena probabilidade, incrementarão suas energias.

Em termos computacionais, a metaheurística SA é uma técnica de busca local probabilística [Souza, 2008]. A SA começa sua busca a partir de uma solução inicial qualquer. O procedimento principal consiste em um laço que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s .

Considerando um problema de minimização, seja Δ a variação de valor da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$. Ocorrem as seguintes situações:

- $\Delta < 0$: Houve uma redução da energia. O método aceita o movimento e a solução vizinha passa a ser a nova solução corrente.
- $\Delta = 0$: Caso de estabilidade, pouco comum na prática. A aceitação do movimento é indiferente.
- $\Delta > 0$: Houve um aumento da energia. A solução vizinha candidata também poderá ser aceita, mas neste caso, com uma probabilidade $e^{-\Delta/T}$, fator de Boltzmann, onde T é um parâmetro do método, chamado de temperatura e que regula a probabilidade de se aceitar soluções de pior custo. A Figura 2.4 demonstra, para $\Delta = 1$, a relação entre a temperatura e o fator de Boltzmann.

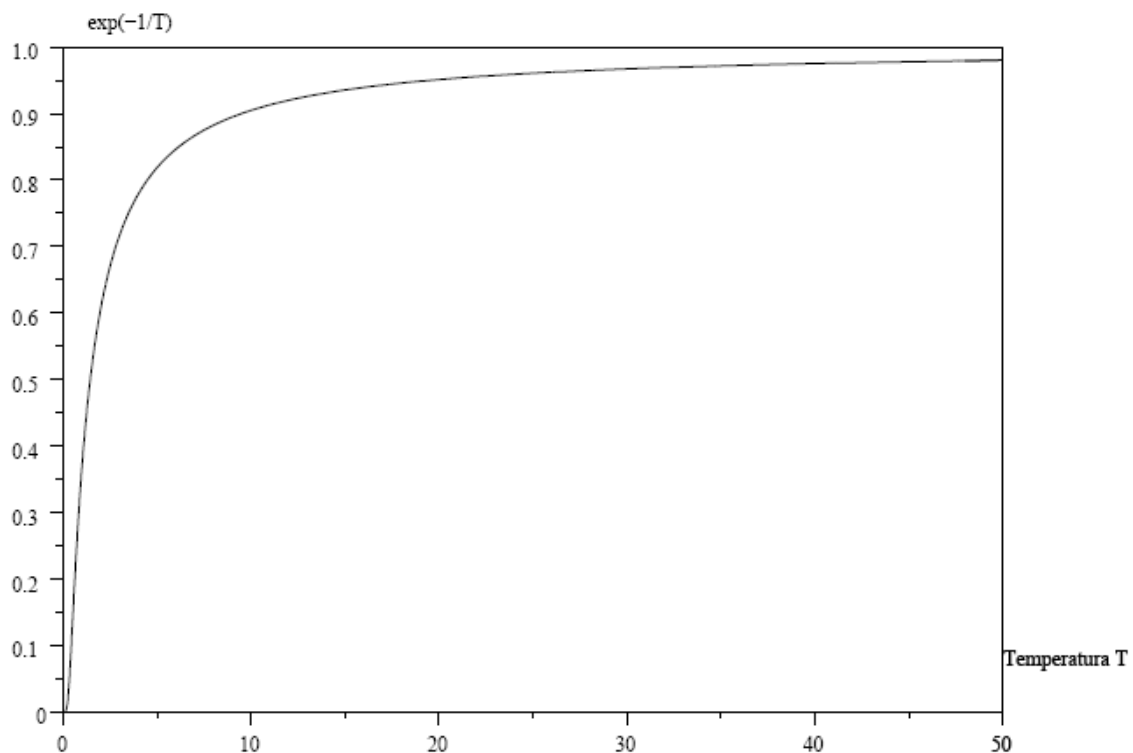


Figura 2.4: Comportamento do fator de Boltzmann em relação à temperatura

A temperatura T assume, inicialmente, um valor elevado T_0 . Após um número fixo de iterações (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento $\alpha[0,1]$, tal que $T_k \leftarrow \alpha \times T_{k-1}$.

Com esse procedimento, no início existe uma chance maior de escape de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de que movimentos de piora sejam aceitos.

O processo é finalizado quando a temperatura chega a um valor próximo a zero e nenhuma solução que piore o valor da solução corrente seja mais aceita. A solução obtida quando o sistema encontra-se nesta situação evidencia o encontro de um mínimo local.

Existem algumas variações entre os algoritmos baseados em SA, que geralmente incluem reaquecimento seguido de um novo processo de resfriamento, utilizado quando a quantidade de movimentos consecutivamente rejeitados é alta. Também é comum trabalhar nas temperaturas mais altas com taxa de resfriamento menor e aumentá-la quando a temperatura reduzir [Souza, 2008].

Os parâmetros de controle do SA são a razão de resfriamento α , o número de iterações para cada temperatura (SA_{\max}) e a temperatura inicial T_0 .

Em teoria, a temperatura final deve ser zero. Entretanto, na prática é suficiente chegar a uma temperatura próxima de zero, devido à precisão limitada da implementação computacional [Torreão, 2008]. Um valor típico é tomar $T_f = 0,001$. Alternativamente, pode-se identificar o congelamento do sistema quando a taxa de aceitação de movimentos apresentar valores abaixo de um nível predeterminado.

Observa-se que, como regra geral, os parâmetros mais adequados para uma dada aplicação do algoritmo só podem ser estabelecidos por experimentação.

O procedimento SA pode ser modelado matematicamente por intermédio da teoria de cadeias de Markov. Existem vários resultados na literatura que, utilizando esse modelo, garantem que esta metaheurística converge para o ótimo global [Hajek, 1988]. O número de iterações necessário para que se o método SA convirja para o ótimo global, no entanto, na maioria dos casos, é computacionalmente proibitivo.

A Figura 2.5 traz o pseudocódigo de um procedimento SA básico.

```

Procedimento SA básico aplicado a um problema de minimização

1  s* <= s;      # Mantém a melhor solução
2  iterT <= 0;   # Numero de iterações para cada T
3  T <= T0;      # Temperatura inicial
4
5  ENQUANTO (T > 0) FAÇA
6      ENQUANTO (iterT < SAmáx) FAÇA
7          iterT <= iterT +1;
8          gere um vizinho s' de s;
9           $\Delta = f(s') - f(s)$ ;
10
11         SE ( $\Delta < 0$ )
12             s <= s' ;
13             SE ( $f(s') < f(s^*)$ ) s* <= s' ;
14         FIM SE
15
16         SE ( $\Delta \geq 0$ )
17             gere aleatoriamente x entre 0 e 1;
18             SE ( $x < e^{-\Delta/T}$ ) s <= s' ;
19         FIM SE
20     FIM ENQUANTO
21
22     T <=  $\alpha \times T$ ;
23     iterT <= 0;
24 FIM ENQUANTO
25
26 Retorne s*; # a melhor solução obtida;

```

Figura 2.5: Pseudocódigo da metaheurística *Simulated Annealing*

2.4 Algoritmos Genéticos

Os Algoritmos Genéticos (GA), do inglês *Genetic Algorithms*, são métodos metaheurísticos de otimização inspirados nos mecanismos de evolução das populações de seres vivos. Foram introduzidos por John Holland [Holland, 1975] e popularizados por um dos seus alunos, David Goldberg [Goldberg, 1989].

Os GA seguem os princípios da seleção natural e da sobrevivência do mais apto, declarados em 1859 por Charles Darwin no livro “A Origem das Espécies” [Lacerda e Carvalho, 1999]. Segundo Darwin, quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.

Em outros termos: o meio ambiente seleciona, a cada geração, os seres vivos mais aptos de uma população para sobrevivência. Como resultado, uma vez que os menos adaptados ao ambiente são eliminados, geralmente, antes de se reproduzir, apenas os mais aptos conseguem gerar descendentes.

Durante a reprodução, ocorrem fenômenos como mutação e recombinação (*crossover*), entre outros, que atuam sobre o material genético armazenado nos cromossomos levando à variabilidade dos seres vivos da população.

Os GA são a metáfora desses fenômenos, herdando, por isso, muitos termos originários da biologia. Listamos a seguir os termos biológicos mais usados nos GA, assim como e sua correspondência computacional:

- **Cromossomo, genoma ou código genético:** Estrutura de dados que codifica uma solução para o problema, ou seja, um simples ponto no espaço de busca.
- **Gene:** Segmento do cromossomo que corresponde (codifica) um parâmetro qualquer do problema.
- **Alelo:** Possível valor que cada gene pode assumir.
- **Aptidão:** Função que avalia um cromossomo a fim de mensurar seu grau de adaptação ao meio. Geralmente a função objetivo do problema é utilizada como função de aptidão.

- **Indivíduo:** Um membro qualquer da população, formado por um cromossomo (codificado ou não) e sua aptidão.
- **População:** Conjunto de indivíduos que coexistem em determinada geração.
- **Clone:** Indivíduos que possuem códigos genéticos completamente iguais.
- **Geração:** Representação numérica de uma dada iteração do algoritmo genético.
- **Genótipo:** Representa a informação contida no cromossomo.
- **Mutação:** Alteração intencional e randômica de um ou mais genes a fim de aumentar a diversificação genética da população.
- **Fenótipo:** Representa o objeto, estrutura ou organismo construído a partir das informações do genótipo.

Os GA iniciam sua busca com a geração de uma população inicial $\{s_1^0, s_2^0, \dots, s_n^0\}$, denominada geração zero ou população em tempo zero.

O procedimento principal é um laço que cria a população $\{s_1^{t+1}, s_2^{t+1}, \dots, s_n^{t+1}\}$ no tempo $t+1$, a partir de uma população no tempo t . Para tal, os indivíduos da geração t passam por um processo de reprodução, que consiste na seleção de indivíduos para recombinação e/ou mutação.

Gerada a nova população em $t+1$, define-se, baseado na função de aptidão, os indivíduos que devem sobreviver, isto é, as n soluções que irão compor a geração $t+1$. Os critérios comumente usados na definição dos sobreviventes são:

1. **Aleatório**, no qual os sobreviventes são escolhidos à sorte;
2. **Roleta**, onde a chance de sobrevivência de cada cromossomo é proporcional ao seu nível de aptidão ou
3. **Misto**, no qual se utiliza uma combinação dos dois métodos anteriores.

Vale salientar que em qualquer desses critérios admite-se a sobrevivência de indivíduos menos aptos tendo-se por objetivo tentar escapar de ótimos locais.

A Figura 2.6 traz o pseudocódigo de um algoritmo genético básico.

Algoritmo Genético Básico

```
1  $t \leftarrow 0$  ;  
2 Gere a população inicial  $P(t)$  ;  
3 Avalie  $P(t)$  ;  
4 ENQUANTO (Critérios de parada não satisfeitos) FAÇA  
5    $t \leftarrow t+1$  ;  
6   gere  $P(t)$  a partir de  $P(t-1)$  ;  
7   Avalie  $P(t)$  ;  
8   Defina a população sobrevivente ;  
9 FIM ENQUANTO
```

Figura 2.6: Pseudocódigo dos Algoritmos Genéticos

Os principais parâmetros de controle dos GA são: o tamanho n da população, o número de gerações, a probabilidade de mutação, a probabilidade de recombinação e o número de iterações sem melhora [Souza, 2008].

2.4.1 Representação de Indivíduos

Um cromossomo p , o qual representa uma solução para o problema, é representado na forma de um vetor com m posições: $p = (x_1, x_2, \dots, x_m)$, onde cada componente x_i representa um gene.

As representações mais conhecidas para os cromossomos são: a representação binária e a representação por inteiros. A característica que diferencia estas representações é o alelo: na primeira, o gene é limitado aos valores '0' ou '1', i.e., $x_i \in \{0,1\}$; enquanto na segunda, o gene pode apresentar qualquer valor inteiro: $x_i \in \mathbb{Z}$.

A representação binária é clássica nos GA, no entanto, existem problemas para os quais é mais conveniente utilizar a representação inteira, por exemplo, o problema do caixeiro viajante [Lacerda e Carvalho, 1999].

2.4.2 Processo de Geração da População Inicial

A escolha do processo de geração da população inicial deve ser tomada levando em consideração dois fatores: uma maior cobertura do espaço de soluções e a minimização de esforços computacionais redundantes.

A geração puramente aleatória tanto pode ocasionar a não representação de algumas regiões do espaço de busca, caso o tamanho escolhido para a população seja muito pequeno, quanto à existência de indivíduos iguais ou muito semelhantes, caso o tamanho da população seja grande demais.

Estes problemas podem ser minimizados através da geração uniforme da população, i.e., com pontos igualmente espaçados, como se preenchessem uma grade no espaço de soluções, vide Figura 2.7.

10000000	00001000
01000000	00000100
00100000	00000010
00010000	00000001

Figura 2.7: Geração uniforme da população inicial

Uma alternativa é gerar apenas a primeira metade da população de forma randômica e a segunda metade a partir da primeira, invertendo-se os bits. Isto garante que cada gene tenha pelo menos um representante na população com os valores '0' e '1'. A Figura 2.8 exemplifica esta abordagem.

1ª metade gerada aleatoriamente	2ª metade inverte os bits da 1ª metade
1011010	0100101
0111011	1000100
0001101	1110010
1100110	0011001

Figura 2.8: Geração da população inicial por inversão

Uma abordagem interessante seria utilizar uma população inicial grande, provendo diversificação do espaço de buscas, e, gradativamente, diminuir o tamanho da população para as gerações subseqüentes, economizando processamento.

Outra técnica comum é o *seeding*, que consiste em inserir na população inicial soluções encontradas por outros métodos, garantindo, assim, que os GA sejam tão ou mais preciso que estes [Lacerda e Carvalho, 1999].

2.4.3 Etapa de Reprodução

A geração da população em tempo $t+1$ dá-se através de uma fase de reprodução, na qual são selecionados indivíduos da geração t para sofrerem operações de recombinação e/ou mutação.

Dentre as formas de seleção de indivíduos para o processo de reprodução, destacam-se a seleção puramente aleatória e a seleção por torneio. No torneio, cada pai é escolhido mediante o seguinte procedimento: são selecionados aleatoriamente k indivíduos da população e apenas aquele que possuir a melhor aptidão é escolhido como pai. Um valor usualmente utilizado é $k=2$, caracterizando o torneio binário (*binary tournament*).

Na operação de recombinação, a qual geralmente ocorre com probabilidade elevada (por exemplo, 80% [Souza, 2008]), os genes de dois (ou mais) cromossomos pais são combinados de forma a gerar filhos (comumente dois). Em cada cromossomo filho haverá um conjunto de genes de cada cromossomo pai.

A operação de mutação, a qual geralmente ocorre com baixa probabilidade (1 a 2%, em geral [Souza, 2008]), consiste em alterar aleatoriamente parte dos genes dos cromossomos.

2.4.3.1 Operador Clássico de Recombinação

A idéia do operador clássico de recombinação (*crossover*) é efetuar cruzamentos entre dois ou mais cromossomos pais e formar cromossomos filhos (*offsprings*) a partir da união de segmentos de genes de cada pai.

Inicialmente são feitos cortes aleatórios nos pais. Por exemplo, considere dois pais e um ponto de corte, representado pelo caractere '|', realizado na parte central dos cromossomos pais:

$$p_1 = (1\ 0\ 0\ 1\ | \ 0\ 1\ 1\ 0) = (p_1^1\ | \ p_1^2)$$

$$p_2 = (1\ 1\ 1\ 1\ | \ 0\ 0\ 0\ 0) = (p_2^1\ | \ p_2^2)$$

A partir dos cortes, são gerados dois filhos, cada qual formado a partir da reunião de partes de cada um dos pais:

$$f_1 = (1\ 0\ 0\ 1\ | \ 0\ 0\ 0\ 0) = (p_1^1\ | \ p_2^2)$$

$$f_2 = (1\ 1\ 1\ 1\ | \ 0\ 1\ 1\ 0) = (p_2^1\ | \ p_1^2)$$

Observe que os cromossomos filhos, apesar de diferentes dos pais, carregam segmentos de informação genética de cada um deles.

2.4.3.2 Operador Clássico de Mutação

O operador de mutação tem como objetivo o escape de ótimos globais. Classicamente, para uma representação binária, consiste em alterar o valor de um ou mais genes de '0' para '1' ou vice-versa. Segue um exemplo de mutação:

$$p = (1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1) \Rightarrow \text{Mutação} \Rightarrow p' = (1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0)$$

Perceba que dois genes foram alterados: o valor de x_3 mudou de '0' para '1' e o valor de x_{11} passou de '1' para '0'.

2.4.4 Critérios de Parada

Dos vários critérios de parada dos GA, os mais comuns são: chegada ao número limite de gerações, convergência da população ou homogeneização dos cromossomos.

A convergência populacional ocorre quando não acontece melhoria significativa da melhor solução por um dado número de gerações. Isto significa que a metaheurística encontrou um ótimo, quer seja local ou global, do qual não consegue escapar. Outra maneira de detectar a convergência populacional é computar, na fase de avaliação da população, o desvio padrão das aptidões dos indivíduos.

A homogeneização dos cromossomos se dá, geralmente, nas gerações mais avançadas dos GA, quando os indivíduos da população possuem entre 90 e 95% dos genes com o mesmo valor [Lacerda e Carvalho, 1999].

2.4.5 Problemas de Convergência

A convergência prematura é um conhecido problema dos GA. Pode ocorrer quando surgem cromossomos com alta aptidão em relação à população, sem que esta possua os cromossomos realmente ótimos; ou quando a população inicial não cobriu suficientemente o espaço de soluções.

Os cromossomos pseudo-ótimos, chamados super-indivíduos, acabam por gerar um excessivo número de filhos, que dominam a população, uma vez esta possui tamanho limitado. Tais cromossomos espalham seus genes por toda a população, enquanto outros genes desaparecem, fenômeno conhecido como *genetic drift*. Como consequência, o método converge para ótimos locais.

A convergência prematura pode ser combatida através da limitação do número de descendentes de cada cromossomo, do aumento da taxa de mutação ou da não inserção de clones na população.

2.5 Fundamentos Sobre Árvore Retilínea Mínima de Steiner

O problema da árvore retilínea mínima de Steiner (RSMTP) destaca-se como um dos problemas fundamentais na automação do roteamento de redes de circuitos eletrônicos, uma vez que uma interconexão de comprimento mínimo possui mínima capacitância total e requer uma mínima quantidade de área [Cabral, 2001].

Define-se árvore retilínea de Steiner como sendo um grafo acíclico e conexo, no qual os vértices (pontos no plano cartesiano) são interligados utilizando apenas segmentos verticais e horizontais, onde é permitida a adição de pontos extras, denominados pontos de Steiner, a fim de diminuir o seu comprimento.

A Figura 2.9 traz exemplos de árvore euclidiana, árvore retilínea e árvore retilínea de Steiner para o mesmo conjunto de pontos.

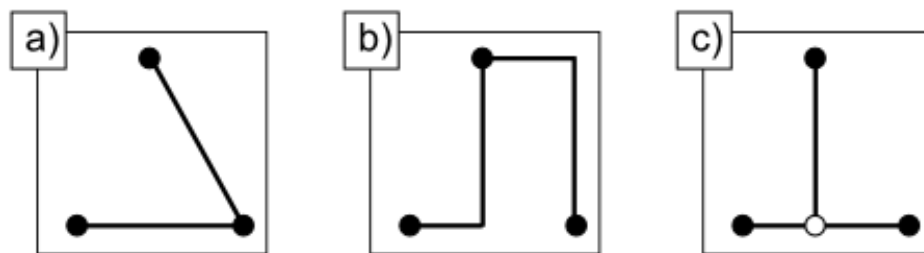


Figura 2.9: (a) Árvore Euclidiana, (b) Árvore Retilínea e (c) Árvore Retilínea de Steiner

Encontrar a árvore retilínea de Steiner de comprimento mínimo (RSMT) para um conjunto qualquer de pontos $P = \{p_1, p_2, \dots, p_n\}$, consiste em selecionar um conjunto de pontos de Steiner $S = \{s_1, s_2, \dots, s_m\}$ tal que $RMST(P \cup S)$ possua o menor custo.

No RSMTP, o custo (comprimento) de uma árvore A , representado por $|A|$, é dado pela soma dos custos das arestas que a compõem. O custo de uma

aresta que conecta os pontos $p_i(x_i, y_i)$ e $p_j(x_j, y_j)$ é aferido através da fórmula $d_1(p_i, p_j) = |x_i - x_j| + |y_i - y_j|$, conhecida como métrica L1 ou distância de Manhattan.

Muitos termos comuns no RSMT são oriundos do jargão do VLSI, tais como: pinos ou terminais, sinônimos de pontos; e *netlist* (ou *net*), que significa um conjunto de terminais.

O rápido avanço das tecnologias VLSI tornou possível o desenvolvimento de circuitos digitais com baixo custo, alto desempenho e elevado número de transistores, culminando na necessidade de realização de roteamento de *nets* cada vez maiores.

2.5.1 Teorema da grade de Hanan

O teorema da grade de Hanan [Hanan, 1966] constitui-se um resultado fundamental das pesquisas sobre o RSMT, seguido pela maioria (senão totalidade) das técnicas de construção de RSMT.

A grade de Hanan é obtida traçando-se linhas verticais e horizontais sobre cada ponto do conjunto de terminais, como demonstra a Figura 2.10. Hanan provou que existe ao menos uma solução ótima para o problema de geração da RSMT que utiliza apenas pontos situados nas intersecções dessas linhas. Posteriormente, Snyder generalizou os resultados de Hanan para qualquer dimensão [Snyder, 1992].

A grade de Hanan propicia uma redução importante no RSMT, permitindo-nos considerar apenas n^2 pontos para a geração da RMST, sendo n terminais e $n^2 - n$ pontos de Steiner.

Apesar da redução do espaço de soluções provida pelo teorema da grade de Hanan, o RSMT mantém-se NP-difícil [Garey e Johnson, 1977]. Zachariasen apresenta um catálogo de problemas que podem ser resolvidos utilizando a grade de Hanan [Zachariasen, 2000].

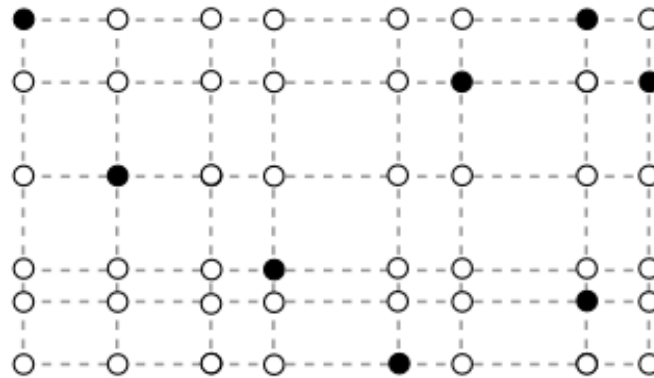


Figura 2.10: Teorema da grade de Hanan

2.5.2 Tipos de pontos de Steiner

Com base na grade de Hanan, os pontos de Steiner podem ser classificados de acordo com o seu grau: *corner-points*, *t-points*, ou *cross-points*, cujos graus são 2,3 e 4 respectivamente. A Figura 2.11 apresenta uma árvore de Steiner composta por terminais, *corner-points* ('a', 'b' e 'd'), *t-points* ('c') e *cross-points* ('e').

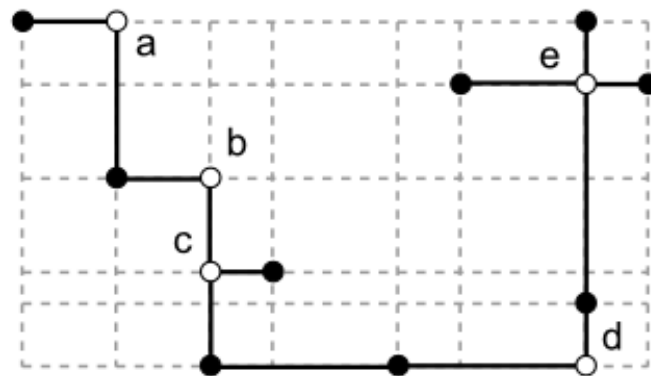


Figura 2.11: Exemplos de pontos de Steiner

Os *corner-points* podem ser removidos sem prejuízo de qualquer árvore de Steiner, pois, por possuírem grau 2, não oferecem nenhuma diminuição no custo desta. A Figura 2.12 demonstra a remoção de um *corner-point* sem impacto no custo total da árvore, visto que $|A| = d_1(a, s) + d_1(s, b) = |B| = d_1(a, b)$.

Por consequência da definição, a RSMT é composta apenas por terminais, *t-points* e *cross-points*. Apesar disto, algumas técnicas permitem o uso de *corner-*

points em fases intermediárias, pois, posteriormente, estes podem se transformar em *t-points* ou *cross-points*.

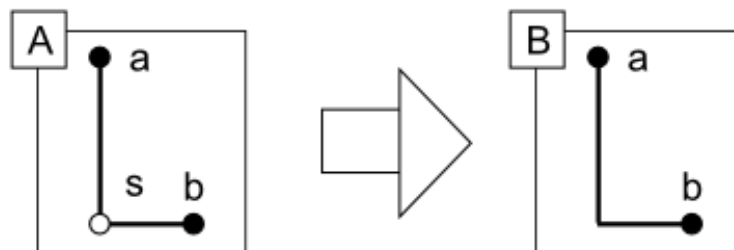


Figura 2.12: Remoção de um *corner-point*

2.5.3 Topologias de árvores de Steiner

Por conta da geometria do RSMT e da grade de Hanan, uma mesma árvore de Steiner pode apresentar diferentes formatos, denominados topologias. Alterar a topologia de uma árvore de Steiner consiste em mudar sua forma mantendo seus terminais, pontos de Steiner e arestas.

A técnica utilizada para alterar a topologia de uma árvore de Steiner é o *flip*: mudança do *corner* utilizado para conectar dois pontos. A Figura 2.13 demonstra quatro topologias diferentes, obtidas através de *flips*, para uma mesma árvore de Steiner.

Em consequência da grande quantidade de topologias existente para uma mesma árvore, o espaço de soluções para o RSMT é composto por muitas regiões planas: vários mínimos locais de mesmo custo e alguns mínimos globais. Esta peculiaridade tem duas implicações:

1. É relativamente fácil encontrar soluções de boa qualidade para o RSMT;
2. Uma vez encontrado um mínimo local, a homogeneidade da vizinhança torna o processo refinamento bastante árduo;

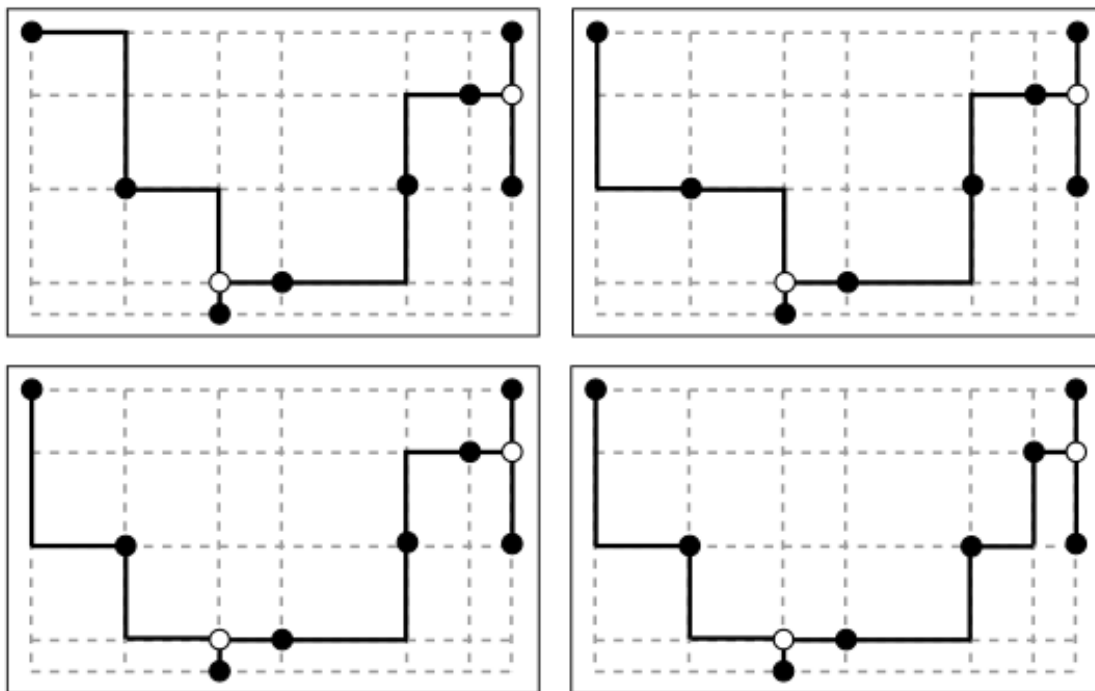


Figura 2.13: Diferentes topologias de uma mesma árvore retilínea de Steiner

2.5.4 Árvores Retilíneas *Full-Steiner*

Uma árvore retilínea *Full-Steiner* (*Rectilinear Full Steiner Tree* - RFST) é uma árvore de Steiner em métrica L1 na qual todo terminal é folha (e toda folha é terminal) e todos os outros vértices – pontos de Steiner – possuem grau três ou quatro [Zachariasen, 2000].

Uma RSMT é uma união de RFSTs e sempre existe uma RSMT na qual cada RFST que a compõe possui uma das duas topologias de Hwang [Hwang, 1976], exibidas na Figura 2.14.

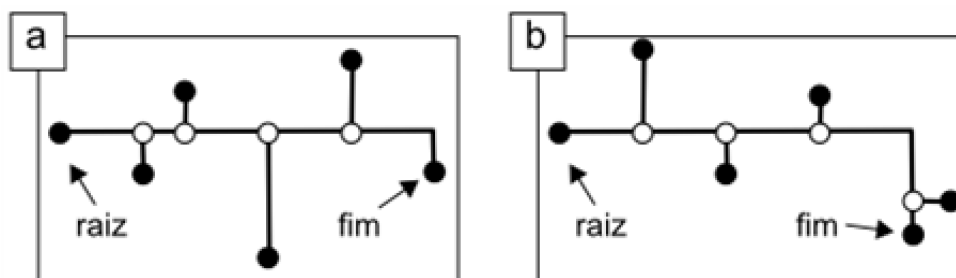


Figura 2.14: Topologias de Hwang para RFSTs

Uma RFST conectando k terminais consiste em um 'L' composto por um ponto inicial (denominado raiz) e um ponto final (denominado fim). A raiz incide sobre a "perna longa" e o fim sobre a "perna curta" do 'L'. A RFST de Hwang tipo (a) possui $k - 2$ segmentos alternados incidentes à perna longa e nenhum segmento incidente à perna curta, enquanto a tipo (b) possui $k - 3$ segmentos alternados incidentes à perna longa e um segmento incidente à perna curta.

Vale salientar que os conceitos de perna longa e perna curta não estão relacionados ao comprimento do segmento, mas sim à quantidade de pontos que incidem sobre este.

As definições de RFST e das topologias de Hwang constituem a base do melhor algoritmo de geração de RSMT da atualidade [Zachariasen, 2000], o GeoSteiner.

2.5.5 Algoritmos de Árvore Geradora Retilínea Mínima

A íntima relação entre o RSMT e a árvore geradora retilínea mínima (RMST, do inglês: *Rectilinear Minimum Spanning Tree*) acarretou que muitas heurísticas de construção de RSMT utilizam a RMST como base.

As primeiras heurísticas de RSMT eram simples melhorias dos algoritmos de RMST, seja através de alterações na fase de geração da RMST ou refinamentos a partir desta.

No entanto, a tarefa de gerar a árvore de cobertura mínima de um grafo completo não é tão simples quanto parece. A complexidade dos algoritmos de Kruskal e Prim é $O(e \log v)$ para um grafo qualquer composto por v vértices e e arestas [Cormen et al., 2001]. Todavia, o número de arestas de um grafo completo é dado por $e = v^2$, aumentando a complexidade para $O(v^2 \log v)$. Na prática essa complexidade pode ser diminuída para $O(v^2)$.

[Hwang, 1976] propôs um algoritmo de geração de RMST em $O(v \log v)$, baseada na construção do diagrama de Voronoi seguida da triangulação de Delaunay. Contudo, por conta da complexidade da geometria utilizada, a implementação dessa abordagem é bastante complexa.

Uma alternativa, de programação mais simples, é realizar uma redução de grau no grafo. [Guibas e Stolfi, 1983] provam que para encontrar uma RMST, basta considerar o grafo no qual cada terminal liga-se apenas com o seu vizinho mais próximo em cada octante (Figura 2.15). Este novo grafo, de grau oito, pode ser construído em $O(v \log v)$ e a computação da RMST deste grafo se dá em $O(v \log v)$ por Kruskal ou Prim.

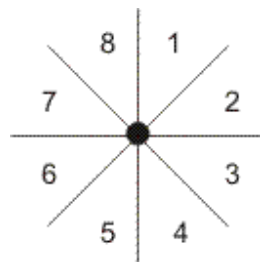


Figura 2.15: Octantes de um ponto qualquer

[Kahng e Mandoiu, 2001] apresenta um estudo comparativo entre três procedimentos de geração de RMST:

1. Uma eficiente implementação do algoritmo de Prim em $O(v^2)$.
2. Uma implementação em $O(v \log v)$ da redução de grau de Guibas e Stolfi seguida do algoritmo de Prim.
3. Uma implementação do algoritmo de Prim com computação dinâmica do vizinho mais próximo em cada octante, proposta por Dr. Lou Scheffer.

A Tabela 2.1 exibe os tempos de execução médios – em segundos – em uma CPU “195MHz SGI Origin 2000”), para instâncias de até 500.000 vértices, de cada um dos algoritmos de RMST estudados em [Kahng e Mandoiu, 2001].

# de Terminais	$O(n^2)$ Prim	Scheffer	Guibas-Stolfi
5	0.000006	0.000400	0.000053
10	0.000024	0.000685	0.000138
50	0.000567	0.003601	0.001118
100	0.002269	0.007959	0.002613
500	0.055323	0.051744	0.017536
1000	0.223079	0.118234	0.038819
5000	5.774400	0.889230	0.243520
10000	23.641100	2.477000	0.531860
50000	N/A	22.685750	3.461500
100000	N/A	75.654200	9.253000
500000	N/A	486.621200	91.634800

Tabela 2.1: Tempo de execução médio dos algoritmos de geração de RMST

Contudo, [Hwang, 1976] provou que as heurísticas de geração de RSMT baseadas em melhorias sobre a RMST têm a qualidade da solução limitada a $\frac{2}{3}|RMST|$ (proporção de Hwang). Visando a superação deste limite, os novos métodos que surgiram utilizam outras abordagens – tal como preceitos geométricos – para resolver o RSMT.

Os algoritmos de RMST, porém, mantêm sua importância, pois, mesmo as técnicas recentes de geração de RSMT se valem do cálculo da RMST. O BI1S (*Batched Iterated First Steiner* [Griffith et al., 1994]), por exemplo, utiliza a RMST para aferir o ganho da inserção de cada ponto de Steiner na solução final.

2.5.6 Half Perimeter Wirelength

O *half perimeter wirelength* (HPWL), metade do perímetro do menor retângulo que contenha todos os pontos de uma *net*, é utilizado comumente pelos posicionadores (*placers*) como estimativa rápida do custo de roteamento final no circuito.

O HPWL de um conjunto de pontos P ordenado por suas coordenadas x e y dá-se em $O(1)$ pela fórmula: $HPWL(P) = (x_{\max} - x_{\min}) + (y_{\max} - y_{\min})$. Caso P não esteja ordenado, localizar x_{\max} , x_{\min} , y_{\max} e y_{\min} é feito em $O(n)$.

O FLUTE [Chu, 2008-A] utiliza o HPWL em sua heurística para *nets* com 10 ou mais terminais.

2.5.7 Aresta e Distância Gargalo

Seja A uma árvore (ou um grafo) composta por um conjunto P de terminais tal que $p_i, p_j \in P$. A aresta de gargalo (*bottleneck edge*) entre p_i e p_j é a aresta de maior custo no(s) caminho(s) entre p_i e p_j em A . A distância de gargalo (*bottleneck distance*) é o custo da aresta de gargalo.

Note que, seja $b(p_i, p_j)$ a distância de gargalo entre p_i e p_j na $RMST(P)$, nenhuma aresta do caminho entre p_i e p_j na $RSMT(P)$ terá comprimento maior que $b(p_i, p_j)$ [Zachariasen, 1997].

O cálculo da aresta de gargalo é importante para a remoção de ciclos em uma árvore. Algumas abordagens para o RSMT – o BGA, por exemplo – são baseadas na adição de pontos de Steiner à RMST e manutenção do comprimento mínimo pela remoção da aresta de gargalo de cada ciclo formado.

2.5.8 Redução de Pontos

As técnicas de redução de pontos constituem uma fase de pré-processamento importante no RSMT. Consistem em remoções de pontos de Steiner da grade de Hanan, tal que exista ao menos uma árvore de Steiner ótima construída com os pontos remanescentes.

[Winter, 1995] discute um conjunto de reduções de pontos e arestas baseado no grau dos vértices e nas distâncias de gargalo do grafo da grade de Hanan. Em média, as reduções propostas por Winter diminuem em 70-80% o número de pontos de Steiner que devem ser considerados.

A Figura 2.16 traz um exemplo das três fases das reduções de Winter. Para uma instância contendo inicialmente oito terminais e cinquenta e seis pontos de Steiner. Aplicadas as reduções, restaram apenas dezesseis pontos de Steiner (diminuição de aproximadamente 72%).

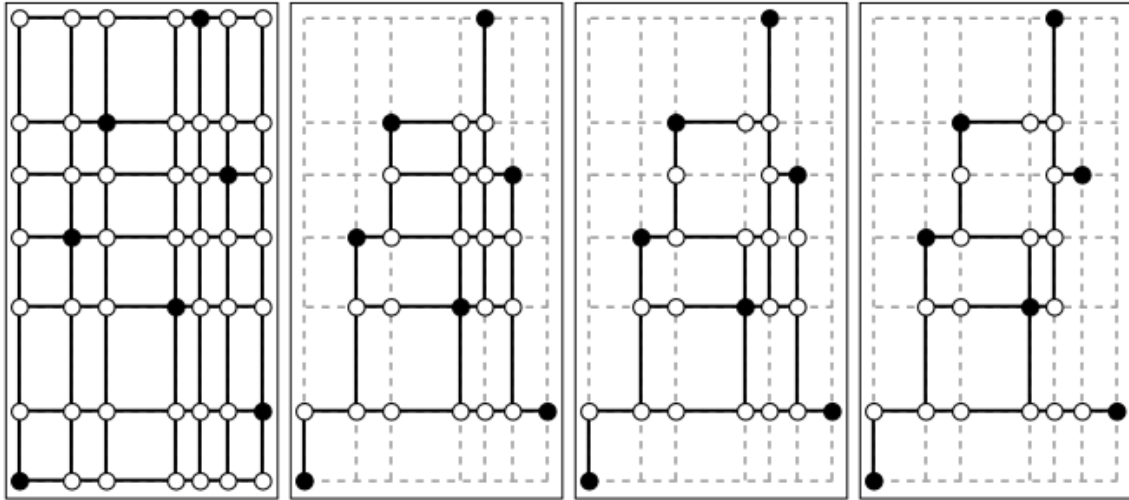


Figura 2.16: Três fases da redução de pontos e arestas de Winter

Outra maneira de reduzir os pontos da árvore de Hanan é através de testes de regiões vazias. O teorema das regiões vazias define que se o segmento uv pertence à RMST, não existe nenhum outro ponto (Steiner ou terminal) dentro das regiões vazias de uv .

[Zachariasen, 1997] demonstra quatro regiões vazias para o RSMT, representadas na Figura 2.17: (a) o diamante, (b) o retângulo, (c) o triângulo e (d) o círculo vazios.

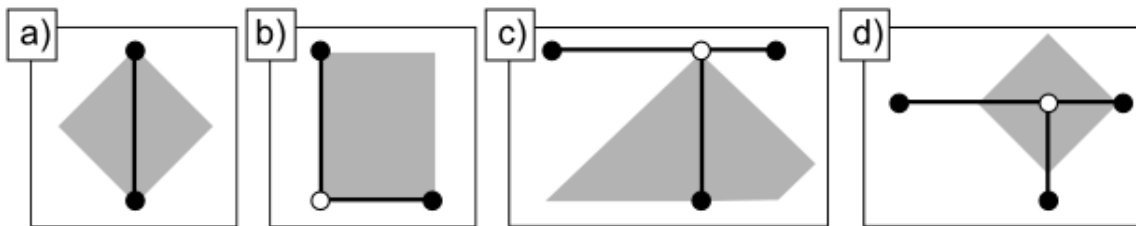


Figura 2.17: Regiões vazias: a) diamante, b) retângulo, c) triângulo e d) círculo

O GeoSteiner [Warme, Winter e Zachariasen, 2009] se vale de testes de diamante e retângulo vazio na fase de geração das RFSTs para eliminar RFSTs que não satisfaçam estas condições e, por consequência, não fazem parte da solução ótima.

CAPÍTULO 3 - O ESTADO DA ARTE

Neste capítulo serão discutidas as técnicas de RSMTTP mais bem sucedidas da atualidade, objetivando a compreensão das suas melhores características para possível utilização no desenvolvimento de novas abordagens.

Existem três classes de técnicas utilizadas para solucionar o problema da menor árvore retilínea de Steiner:

- Abordagens exatas: técnicas utilizadas para resolver instâncias que contêm número reduzido de pontos.
- Heurísticas semi-ótimas: técnicas utilizadas para resolver instâncias maiores do problema, com ênfase à aplicação na fase de roteamento de chips da classe FPGA (*Field Programmable Gate Array*).
- Heurísticas rápidas: técnicas utilizadas nas fases iniciais do projeto de *chips* – como a síntese ou o posicionamento -- para estimar o comprimento final do roteamento.

A Figura 3.1 exibe a classificação das abordagens estudadas neste capítulo segundo as classes apresentadas.

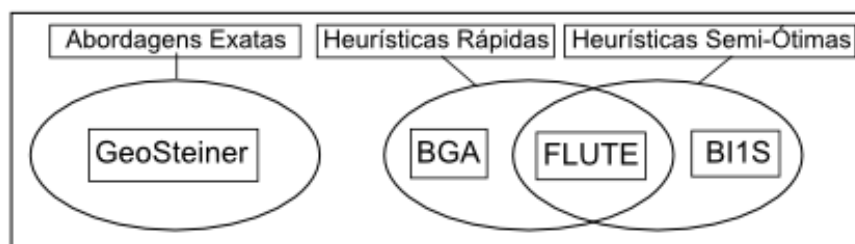


Figura 3.1: Classes de Técnicas para o RSMTTP

São utilizados dois métodos para qualificar uma heurística H de RSMTTP:

- A melhoria média propiciada pelas soluções de H sobre as respectivas RMSTs, dada por $\delta(H) = \frac{(|RMST| - |H|)}{|RMST|}$;

- A diferença média entre as soluções de H e as respectivas soluções ótimas, dada por $FFO(H) = (|H| - |\text{Ótimo}|) / |H|$;

3.1 BI1S

O *Batched Iterated 1-Steiner* (BI1S) consiste em uma implementação prática da heurística *Iterated 1-Steiner* (I1S), utilizando um novo esquema de manutenção dinâmica da RMST e o conceito de liberdade entre pontos de Steiner.

O I1S [Kahng e Robins, 1992] foi proposto como alternativa às heurísticas baseadas na construção de RMST, apresentando desempenho que supera a proporção de Hwang.

A idéia central do I1S é encontrar, iterativamente, o ponto de Steiner que cause o maior decréscimo no custo da solução, dito 1-Steiner, e adicioná-lo até que nenhuma melhoria seja possível. Sejam P e S dois conjuntos de pontos, a melhoria de S sobre P é definida como: $\Delta RMST(P, S) = RMST(P) - RMST(P \cup S)$. Então, o ponto 1-Steiner $s \in S$ maximiza $\Delta RMST(P, \{s\}) > 0$.

Apesar de uma RSMT conter no máximo $n - 2$ pontos de Steiner [Kahng e Robins, 1992], o I1S pode adicionar mais do que $n - 2$ pontos. Por isso, a cada iteração são eliminados os pontos de Steiner de grau dois ou menos. A Figura 3.2 ilustra uma execução simples do I1S para uma instância composta por quatro terminais.

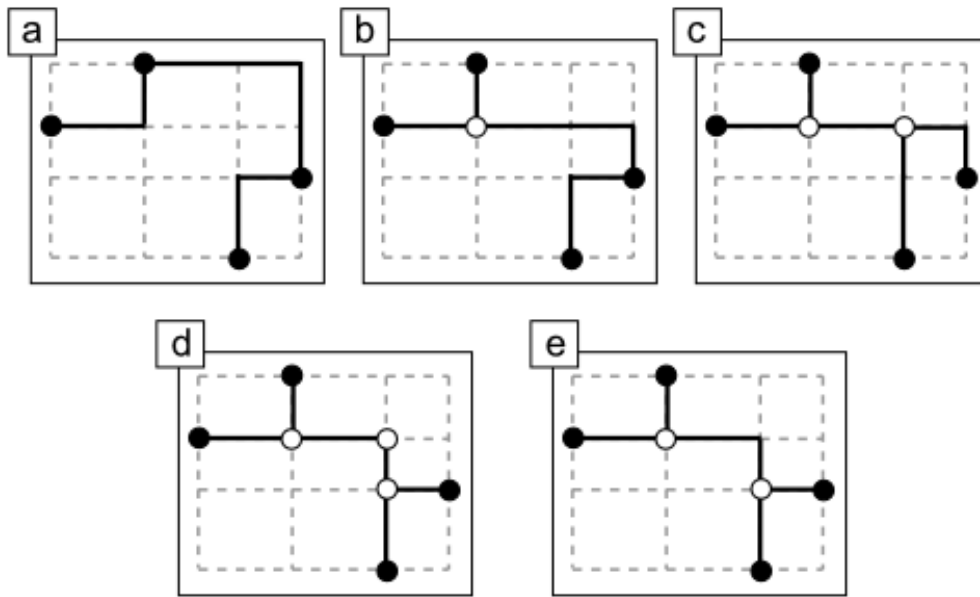


Figura 3.2: Execução simples do I1S

Atente que na Figura 3.2(d) existem três pontos de Steiner, ultrapassando o limite de $n - 2 = 2$ permitido por esta instância. Por isso, na Figura 3.2(e) há a remoção de um ponto de Steiner de grau dois.

Apesar de apresentar uma qualidade interessante: $\delta(I1S) \cong 11\%$ e $FPO(I1S) \cong 0.5\%$, o I1S possui uma complexidade computacional muito alta: $O(n^4 \log n)$.

O BI1S [Griffith et al., 1994], consiste em uma melhoria na complexidade computacional do I1S, sem afetar sua qualidade, utilizando o conceito de independência entre pontos e um esquema de manutenção dinâmica da árvore geradora mínima em distância retilínea.

As inovações apresentadas pelo BI1S, reduzindo a complexidade do I1S para $O(n^3)$, fizeram esta heurística ostentar a melhor qualidade média entre as técnicas de RSMT por muitos anos, sendo superada apenas em 2004 pelo FLUTE [Chu, 2008-A].

3.1.1 Independência entre pontos

Dois pontos a e b são ditos independentes se a inserção do ponto a na $RMST(P \cup \{b\})$ não diminuir o $\Delta RMST(P, \{b\})$. A idéia principal contida no BI1S consiste em, a cada interação, ao invés de adicionar o 1-Steiner, adicionar o conjunto S de pontos independentes de Steiner que maximize $\Delta RMST(P, S)$.

Formalmente, um conjunto S de pontos de Steiner é dito independente se satisfaz a condição: $\Delta RMST(P, S) \geq \sum_{x \in S} \Delta RMST(P, \{x\})$.

A tarefa de encontrar o melhor conjunto de pontos de Steiner independentes herda a complexidade NP-difícil do RSMTP. Contudo, uma implementação gulosa apresenta bons resultados na prática [Griffith et al., 1994].

Uma vez selecionado o (aproximadamente) melhor conjunto de pontos independentes de Steiner, este é inserido na solução e inicia-se uma nova iteração do BI1S. Este processo é repetido até que uma iteração falhe em adicionar pontos de Steiner.

O número de iterações executadas pelo BI1S – detalhadas na Tabela 3.1 – é uma constante que independe do tamanho da instância e apresenta um valor pequeno (em média menor que três).

n	Pontos de Steiner			Iterações		
	min	média	max	min	média	max
3	0	0,90	1	1	1,90	2
4	0	1,53	2	1	2,18	3
5	1	2,25	3	2	2,33	4
7	1	3,63	6	2	2,60	5
10	2	5,63	8	2	2,92	7
14	5	8,22	11	2	3,18	6
20	9	12,14	15	2	3,26	5
30	15	18,99	22	3	3,53	6
50	29	32,86	35	3	4,14	6

Tabela 3.1: Número de iterações e número de pontos adicionados por iteração no BI1S

3.1.2 Manutenção dinâmica da RMST

[Griffith et al., 1994] propõe um novo esquema de manutenção dinâmica da RMST com o intuito de agilizar o cálculo da melhoria que a adição de um conjunto de pontos causa sobre esta (base do BI1S).

Uma vez calculada a árvore geradora mínima para um conjunto de pontos P , o acréscimo de um único ponto x a P produz apenas um pequeno e constante número de alterações entre $RMST(P)$ e $RMST(P \cup \{x\})$.

Parte-se da observação de que para a manutenção dinâmica da RMST, é suficiente considerar apenas quatro vizinhos de cada novo ponto de Steiner adicionado, o mais próximo para cada região definida por duas retas orientadas a 45° e -45° (Figura 3.3), dita partição diagonal de x .

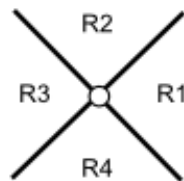


Figura 3.3: Quadrantes definidos pela partição diagonal de um ponto

Então, para manter a RMST em consequência da adição de um novo ponto x , basta conectar x ao seu vizinho mais próximo em cada região de sua partição diagonal e remover a aresta de maior comprimento de cada ciclo formado. A Figura 3.4 demonstra a inclusão de um novo ponto em uma árvore com manutenção dinâmica da RMST.

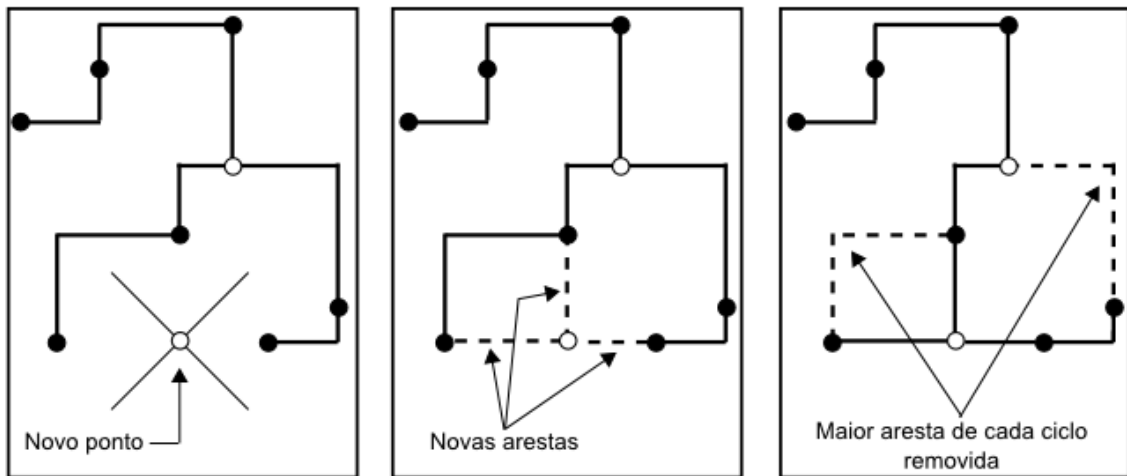


Figura 3.4: Manutenção dinâmica da RMST

A inclusão da técnica de manutenção dinâmica de RMST reduz a complexidade do BI1S para $O(Kn^3)$, onde K é o número de iterações realizadas pela heurística. Como o número de iterações é constante, tem-se que a complexidade do BI1S é $O(n^3)$.

3.1.3 Outras melhorias no BI1S

Outras melhorias na velocidade do BI1S são advindas de técnicas de redução de pontos. Madoiu propõe uma melhoria significativa no tempo de execução do BI1S baseada no teste do retângulo vazio [Mandoiu, 1999].

3.2 BGA

O BGA (*Batched Greedy Algorithm*) [Kahng, Mandoiu e Zelikovsky, 2003] foi proposto com o intuito de resolver instâncias muito grandes do RSMT em tempo hábil e sem perda de qualidade. Para tal, apresenta uma heurística em $O(n \log^2 n)$ que utiliza $O(n)$ memória. Este tempo de execução caracteriza o BGA como primeira heurística sub-quadrática para o RSMT.

A eficiência do BGA é proveniente de três conceitos chave:

- Uma combinação da implementação do GTCA (*Greedy Triple Contraction Algorithm*) [Zelikovsky, 1992] com a idéia do conjunto de pontos independentes B1S.
- Um novo método de divisão-e-conquista para calcular o conjunto de triplas requerido pelo GTCA.
- Uma estrutura de dados linear que possibilita a localização da aresta de gargalo em $O(\log n)$, após um pré-processamento em $O(n \log n)$.

O GTCA consiste em computar a (aproximadamente) menor árvore 3-restrita para um conjunto de pontos, sendo uma árvore k -restrita composta por RFSTs (denominadas componentes) contendo no máximo k terminais. A Figura 3.5 demonstra uma árvore 3-restrita.

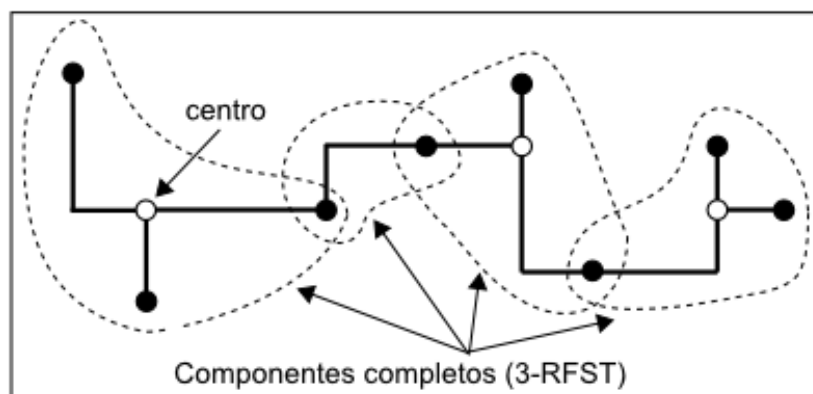


Figura 3.5: Árvore 3-restrita

A cada iteração, o GTCA seleciona gulosamente uma tripla τ , onde tripla é uma RSMT de uma net de três pontos – i.e. um componente 3-restrito – que reduza o custo da RMST, e adiciona-a à solução. Para tal, é necessária a computação de todas as triplas possíveis e seus ganhos, tal que o ganho de uma tripla τ , denotado $\Delta RMST(P, \tau)$, é calculado de maneira análoga ao cálculo do ganho na inserção de um ponto de Steiner no I1S.

O número de triplas necessário, porém, é pequeno, pois se prescindem as triplas que não diminuem o custo da RMST ($\Delta RMST(P, \tau) \leq 0$) e as que não são vazias [Fößmeier, Kaufmann e Zelikovsky, 1997]. Uma tripla τ é vazia se o fecho convexo desta não contiver nenhum outro terminal.

O BGA utiliza-se de uma abordagem de divisão-e-conquista que computa em $O(n \log n)$ um conjunto contendo todas as $O(n \log n)$ triplas vazias.

Uma vez obtido o conjunto de todas as triplas vazias, o BGA seleciona, a cada iteração, o (aproximadamente) melhor conjunto de triplas independentes e adiciona à solução. O conceito de independência entre triplas é análogo ao de independência de pontos do BI1S.

É fácil perceber que a computação do ganho de cada tripla implica em encontrar e eliminar a aresta gargalo de cada ciclo formado sua inserção. O *hierarchical greedy preprocessing* (HGP), computa, para uma dada árvore com n terminais, dois vetores auxiliares, *parent* e *edge*, com no máximo $2n - 1$ elementos cada. Usando estes vetores, a aresta de gargalo entre dois terminais quaisquer é localizada em $O(\log n)$.

Dado um conjunto de arestas ordenado ascendentemente de acordo com seu custo, o procedimento HGP – descrito na Figura 3.6 – calcula os vetores *edge* e *parent* em $O(n)$. A complexidade do HGP é dominada, portanto, pela ordenação $O(n \log n)$ das arestas.

Entrada:	Árvore $T(V=\text{vértices}, E=\text{arestas})$ com n vértices
Saída:	Arrays $\text{edge}(i)$ e $\text{parent}(i)$, $i=1, \dots, 2n-1$
1.	Ordene as arestas em ordem ascendente de custo
2.	Inicialização $\text{next} \leftarrow n$ PARA cada $i=1, 2, \dots, 2n-1$ FAÇA $\text{parent}(i) \leftarrow \text{NULL}$ $\text{edge}(i) \leftarrow \text{NULL}$
3.	PARA cada aresta $e(i) = (u, v)$, $i=1, \dots, n-1$, FAÇA ENQUANTO $u \neq v$ E $\text{parent}(u) \neq \text{NULL}$ E $\text{parent}(v) \neq \text{NULL}$ FAÇA $u \leftarrow \text{parent}(u)$ $v \leftarrow \text{parent}(v)$ SE $\text{parent}(u) = \text{parent}(v) = \text{NULL}$, ENTÃO $\text{next} \leftarrow \text{next} + 1$ $\text{parent}(u) \leftarrow \text{parent}(v) \leftarrow \text{next}$ $\text{edge}(u) \leftarrow \text{edge}(v) \leftarrow i$ SE $\text{parent}(u) = \text{NULL}$ E $\text{parent}(v) \neq \text{NULL}$, ENTÃO $\text{parent}(u) \leftarrow \text{parent}(v)$ $\text{edge}(u) \leftarrow i$ SE $\text{parent}(u) \neq \text{NULL}$ E $\text{parent}(v) = \text{NULL}$, ENTÃO $\text{parent}(v) \leftarrow \text{parent}(u)$ $\text{edge}(v) \leftarrow i$
4.	Retorne os arrays parent e edge

Figura 3.6: Pseudocódigo do HGP [Kahng, Mandoiu e Zelikovsky, 2003]

Computados os vetores edge e parent , o procedimento da Figura 3.7 encontra a aresta de gargalo no caminho entre dois terminais $O(\log n)$.

Entrada:	Árvore conjunto E de arestas ordenado; arrays edge e parent ; e os terminais u e v
Saída:	Aresta de maior custo no caminho entre u e v
1.	$\text{index} \leftarrow -\infty$
2.	ENQUANTO $u \neq v$ FAÇA $\text{index} \leftarrow \max\{\text{index}, \text{edge}(u), \text{edge}(v)\}$ $u \leftarrow \text{parent}(u)$ $v \leftarrow \text{parent}(v)$
3.	Retorne $e(\text{index})$

Figura 3.7: Rotina de computação da aresta de gargalo do caminho entre dois terminais u e v

A inclusão do HGP e do conceito do conjunto de triplas independentes fazem cada iteração do BGA possuir uma complexidade computacional de $O(n \log^2 n)$. Como, na prática, assim como no BI1S, o número de iterações é pequeno, esta é a complexidade do BGA.

Apesar de apresentar, em média, soluções de qualidade um pouco inferior se comparado ao BI1S ($\delta(BGA) \cong 11\%$ e $FFO(BGA) \cong 0.6\%$), o BGA caracteriza-se, por conta de seu tempo de execução e da utilização de estrutura de memória linear, como a heurística mais escalonável para o RSMT, devido ao seu tempo de execução e pela utilização de estrutura de memória linear. Ela é capaz de encontrar boas soluções para instâncias compostas por milhares de terminais.

3.3 FLUTE

O *Fast Lookup Table Based Wirelength Estimation Technique* (FLUTE), proposta inicialmente por Chris Chu em 2004, consiste em uma heurística de geração de RSMT rápida e de excelentes resultados para instâncias com pequeno número de pontos. Estudos empíricos comprovam o comportamento ótimo do FLUTE para *nets* com nove ou menos terminais [Chu, 2008-A].

A heurística baseia-se na pré-computação de topologias potencialmente ótimas, relacionadas à posição relativa entre os pontos da instância. Estas topologias são guardadas em tabelas (*Lookup Tables*), e a computação da RSMT reduz-se à minimização do custo destas topologias pré-tabeladas.

O cálculo das topologias possíveis para cada instância depende dos conceitos de sequência vertical, distância marginal e vetor de coeficientes.

3.3.1 Sequência Vertical

Dada uma instância P qualquer, composta por n pontos (x_i, y_i) , para $1 \leq i \leq n$, ordenados ascendentemente em respeito à sua coordenada x , assumamos $x_1 \leq x_2 \leq \dots \leq x_n$. Define-se sequência vertical $s_1 s_2 \dots s_n$ como a lista de índices dos

pontos de P ordenados de acordo com a coordenada y . A Figura 3.8 exibe uma instância composta por quatro terminais cuja sequência vertical é 3142.

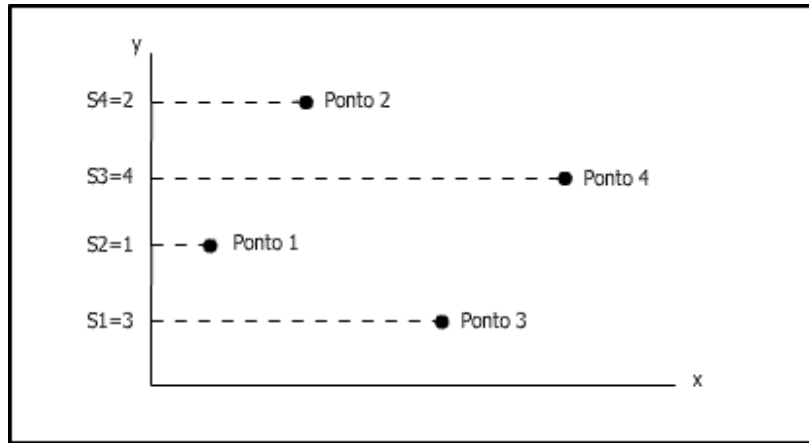


Figura 3.8: Sequência Vertical

Em outros termos, a sequência vertical de uma instância classifica-a de acordo com a posição relativa entre seus pontos. Instâncias que compartilham a mesma sequência vertical compõem um grupo cujas RSMT podem ser construídas obedecendo a um pequeno número de topologias pré-definidas.

3.3.2 Distância Marginal

Define-se como distância marginal horizontal (vertical), a distância entre duas linhas horizontais (verticais) adjacentes na grade de Hanan. Denotam-se todas as distâncias entre margens horizontais $h_i = x_{i+1} - x_i$ e verticais $v_i = y_{s_{i+1}} - y_{s_i}$, para $1 \leq i \leq n$. Estas definições podem ser observadas na Figura 3.9.

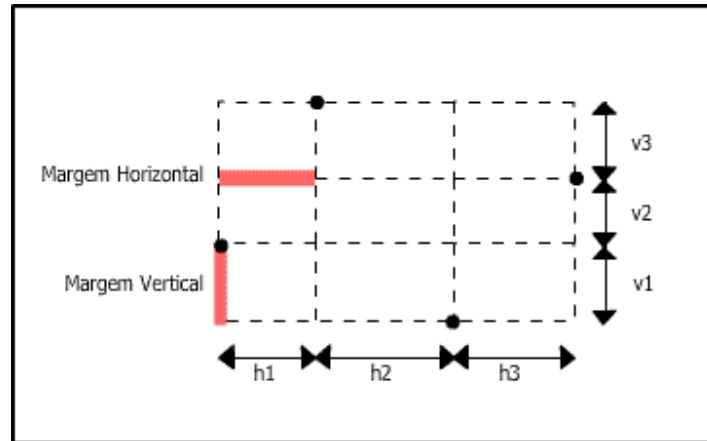


Figura 3.9: Distância marginal

3.3.3 Vetores de Coeficientes Potencialmente Ótimos

Note que qualquer solução para o RSMTF pode ser escrita como uma combinação linear de distâncias marginais cujos coeficientes são números inteiros positivos. Por exemplo, para a instância da Figura 3.8, as três possíveis soluções mostradas na Figura 3.10 (a), (b) e (c) podem ser escritas nas formas $h_1 + 2h_2 + h_3 + v_1 + v_2 + 2v_3$, $h_1 + h_2 + h_3 + v_1 + 2v_2 + 3v_3$ e $h_1 + 2h_2 + h_3 + v_1 + v_2 + v_3$, respectivamente. Por simplicidade, é utilizado o vetor dos coeficientes das distâncias marginais como notação. Para a Figura 3.10 (a) temos (1,2,1,1,1,2).

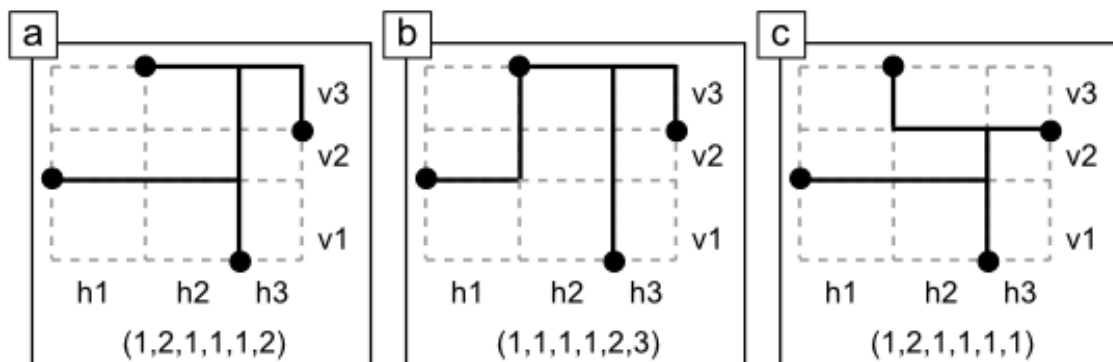


Figura 3.10: Vetores de Coeficientes

Para cada grupo – instâncias que compartilham mesma sequência vertical – é calculado o conjunto de vetores de coeficientes que podem levar à solução ótima (POWV, do inglês *Potentially Optimal Wirelength Vector*). Muitos vetores são

redundantes, pois apresentam valores maiores ou iguais do que outro vetor em todos os coeficientes. Por exemplo, o vetor de coeficientes da Figura 3.10 (a) (1,2,1,1,1,2) pode ser ignorado, pois o vetor de coeficientes da Figura 3.10 (c) (1,2,1,1,1,1) é sempre menor do que este.

Uma instância contendo n terminais possui exatamente $n!$ grupos (sequências verticais), e cada grupo apresenta um pequeno número de POWVs, como demonstra a Tabela 3.2. Por conta disto, é possível pré-calculer todos os POWVs possíveis para pequenas instâncias.

n	n!	# POWVs por grupo		
		min	média	max
2	2	1	1,000	1
3	6	1	1,000	1
4	24	1	1,667	2
5	120	1	2,467	3
6	720	1	4,433	8
7	5040	1	7,932	15
8	40320	1	15,803	34
9	362880	1	30,039	79

Tabela 3.2: Relação de POWVs por grupo

Uma vez de posse do conjunto de POWVs e dos custos das distâncias marginais para a instância apresentada, a computação da RSMT consiste apenas em algumas somas e multiplicações com o objetivo de encontrar o POWV de menor custo.

Utilizando técnicas complexas, Chu conseguiu gerar todos os POWVs para instâncias compostas por nove ou menos terminais, garantindo o comportamento ótimo do FLUTE para estas. A partir daí, a persistência é fator proibitivo para computação, tabelamento e acesso aos POWVs.

3.3.4 FLUTE para Instâncias com Dez ou Mais Terminais

Para instâncias compostas por dez ou mais pontos, o tempo necessário para gerar as tabelas de POWVs e a quantidade de memória necessária para guardá-las tornam o FLUTE impraticável. Para estes problemas, é utilizada uma

técnica de quebra da instância, dividindo-a em sub-instâncias de dois a nove terminais.

Para quebrar uma instância, escolhe-se um ponto, dito ponto de quebra, e uma direção (horizontal ou vertical), separando as instâncias em duas, de acordo com as coordenadas do ponto de quebra e a direção escolhida. O ponto de quebra é incluído em ambas as sub-instâncias, que são solucionadas independentemente. A quebra é, então, realizada recursivamente até que nenhuma sub-instância possua mais que nove pontos. A Figura 3.11 (a) demonstra uma quebra de instância em direção horizontal, enquanto a Figura 3.11 (b) exibe a RSMT gerada para cada sub-instância.

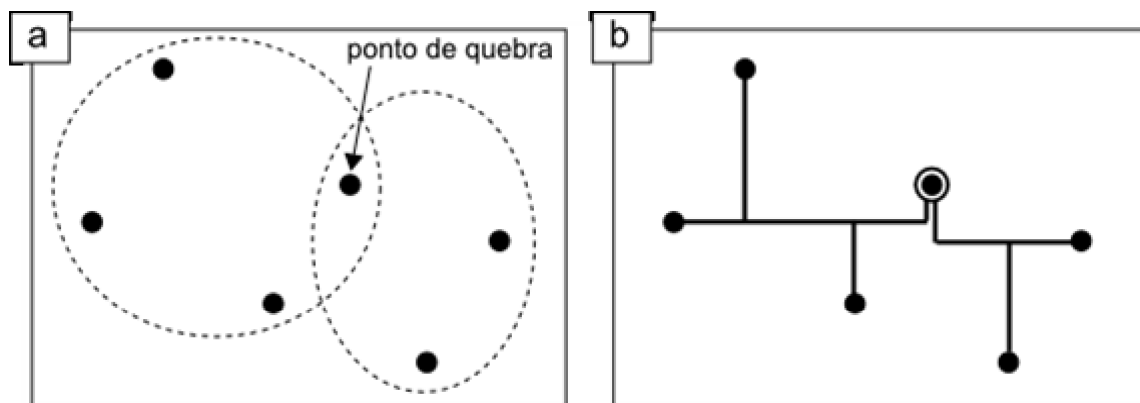


Figura 3.11: Quebra de Instância

A escolha da direção e do ponto de quebra afeta diretamente a qualidade da solução, ocasionando a perda do comportamento ótimo do FLUTE para instâncias com dez ou mais terminais.

A primeira versão da heurística (FLUTE 1.0 [Chu, 2004]), utiliza o HPWL como heurística subordinada para estimar o melhor ponto de quebra em cada direção. A técnica é, então, executada para cada ponto/direção selecionados, retornando apenas o melhor resultado.

No FLUTE 2.0 [Chu e Wong, 2005], são apresentadas três heurísticas de quebra de instâncias e o parâmetro A , definido como número de maneiras em que uma instância será quebrada. Apesar de aumentar substancialmente o tempo de execução do algoritmo, propicia ao usuário a escolha entre qualidade e tempo de

execução do programa. A Tabela 3.3 exibe um comparativo entre qualidade da solução e o valor do parâmetro A , onde se constatou o melhor custo-benefício para $A=3$. Nas versões mais recentes do FLUTE, são recomendados $A=6$ e $A=12$ para uma maior qualidade na solução.

Parâmetro A	% FFO	Tempo de Execução	
		(s)	Normalizado
$A=1$	0,330	5,04	0,61
$A=2$	0,151	6,16	0,74
$A=3$	0,070	8,31	1,00
$A=4$	0,039	12,10	1,46
$A=5$	0,026	18,36	2,21
$A=6$	0,020	29,60	3,56
$A=7$	0,016	51,38	6,18
$A=8$	0,013	96,35	11,59
$A=9$	0,012	190,67	22,94

Tabela 3.3: Parâmetro A

Recentemente, Chu apresentou o FLUTE 3.0 [Chu, 2008-B], com melhorias significativas em relação às versões anteriores. As maiores contribuições da nova versão são a técnica de quebra das instâncias baseada em RMST e as heurísticas de junção FLUTE-MR (*Merge Redundant*) e FLUTE-AM (*Aggressive Merge*).

3.3.5 Complexidade Computacional e Resultados

Por possuir os POWVs já pré-computados, o FLUTE possui um tempo de execução sub-quadrático, no qual se pode, através do parâmetro A , optar por uma solução melhor em troca de um maior custo computacional. A sua complexidade computacional é $O(A!n \log n)$.

Dentre as heurísticas de RSMT contemporâneas, o FLUTE 3.0 com $A=12$ é a técnica que apresenta as soluções mais próximas do ótimo ($FFO(FLUTE12) \cong 0.3\%$).