COPPE/UFRJ

ALGORITMOS DE OTIMIZAÇÃO APLICADOS AO PROBLEMA DE STEINER EM N DIMENSÕES

Vinícius Leal do Forte

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Nelson Maculan Filho
Flávio Marcelo Tavares Montenegro

Rio de Janeiro
Fevereiro de 2010

ALGORITMOS DE OTIMIZAÇÃO APLICADOS AO PROBLEMA DE STEINER EM N DIMENSÕES

Vinícius Leal do Forte

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D.Habil.

Prof. Flávio Marcelo Tavares Montenegro, D.Sc.

Prof. José André de Moura Brito, D.Sc.

Prof. Paulo Roberto Oliveira, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2010

Forte, Vinícius Leal do

Algoritmos de Otimização Aplicados ao Problema de Steiner em n dimensões/Vinícius Leal do Forte. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIII, 101 p.: il.; 29, 7cm.

Orientadores: Nelson Maculan Filho

Flávio Marcelo Tavares Montenegro

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 85 – 90.

1. Problema de Steiner Euclidiano. 2. Heurísticas e Metaheurísticas. 3. Otimização. I. Maculan Filho, Nelson *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Agradeço ao meus pais pelo amor, carinho e apoio, essenciais para que pudesse atingir os meus objetivos.

Ao professor Nelson Maculan, por sua orientação e por ter compartilhado de seu vasto conhecimento com este aluno.

Ao professor Flávio Montenegro, por sua dedicação ímpar a elaboração dessa dissertação de mestrado.

Ao professor José André Brito, pelas idéias e observações que contribuíram para melhorar a qualidade dessa dissertação.

Aos membros da minha banca de dissertação, pelas sugestões que foram de extrema importância para este trabalho.

Aos professores e funcionários do PESC, em especial aos professores Paulo Roberto, Abílio Lucena, Adilson Xavier e Felipe França.

Aos amigos do PESC, que com comentários e discuções ajudaram a enriquecer esta dissertação, em especial a Diana Sasaki pelo carinho e atenção.

Aos amigos de longa data, pelo apoio, estímulo e compreenção.

Agradeço a CAPES pela bolsa de mestrado concedida.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMOS DE OTIMIZAÇÃO APLICADOS AO PROBLEMA DE STEINER EM N DIMENSÕES

Vinícius Leal do Forte

Fevereiro/2010

Orientadores: Nelson Maculan Filho
Flávio Marcelo Tavares Montenegro

Programa: Engenharia de Sistemas e Computação

O Problema de Steiner Euclidiano em n dimensões consiste em, dado um conjunto de pontos em um espaço euclidiano n-dimensional, encontrar a rede de tamanho mínimo que os interconecta, podendo ser adicionados pontos extras à rede. Existem diversos algoritmos exatos e heurísticos para a resolução do PSE para $n = 2$. No entanto, poucos algoritmos foram desenvolvidos para $n \geq 3$. Nesta disertação, apresentamos novos algoritmos para o problema n-dimensional baseados na metaheurística Busca Local Iterativa (ILS) e em heurísticas de relaxação dinâmica anteriormente propostas na literatura. Resultados computacionais são apresentados e utilizados para analisar a qualidade das soluções produzidas pelos algoritmos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

OPTIMIZATION ALGORITHMS APPLIED TO THE STEINER'S PROBLEM IN N DIMENSIONS

Vinícius Leal do Forte

February/2010

Advisors: Nelson Maculan Filho
Flávio Marcelo Tavares Montenegro

Department: Systems Engineering and Computer Science

The Euclidean Steiner Tree problem in n dimensions consists in, given a set of points in an n-dimesional Euclidean space, to find a network which spans these points with minimal length, being allowed to add extra points to the network. There exists several exact and heuristic algorithms to solve the PSE when $n = 2$. However, only a few algorithms were developed for $n \geq 3$. In this dissertation, we present new algorithms for the n-dimensional problem which are based on the Iterated Local Search (ILS) metaheuristic and on a dynamic relaxation heuristic previously proposed in the literature. Computational results are presented and used to analyze the quality of the solutions produced by the algorithms.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O presente trabalho tem como objetivo o estudo e a proposta de um conjunto de algoritmos heurísticos para a resolução do Problema de Steiner Euclidiano (PSE) em n dimensões. Este problema é definido como: Dados p pontos em um espaço euclidiano n-dimensional, encontrar a rede de menor tamanho que os interconecta, podendo ser acrescentados pontos extras, denominados pontos de Steiner.

Este problema é considerado NP-difícil [2] e reúne tanto características de um problema contínuo quanto de um problema discreto. Em função da complexidade desse problema, a utilização de algoritmos heurísticos é pertinente. Em geral, tais algoritmos produzem soluções de boa qualidade, em um tempo computacional factível, mas sem garantia de otimalidade.

Até o presente momento, diversas heurísticas e algoritmos exatos foram propostos para este problema, quando considerados pontos pertencentes ao plano. No entanto, para dimensões maiores que 2, o problema conta com apenas um algoritmo exato, o qual restringe a resolução do PSE para poucos pontos. Por fim, acrescenta-se também, alguns poucos algoritmos heurísticos, os quais produzem soluções de boa qualidade para problemas com até 250 pontos dados.

Essa dissertação está organizada da seguite forma: No próximo capítulo apresentaremos um pouco da história do Problema de Steiner Euclidiano, as definições associadas a este problema e suas propriedades. Também apresentaremos algumas aplicações desse problema e uma breve descrição dos principais algoritmos propostos na literatura. Finalizamos o capítulo descrevendo o algoritmo exato para resolução do PSE em $\Re^n$.

Em seguida, no capítulo 3, serão apresentados os procedimentos contidos nos algoritmos de Busca Local Iterativa (ILS) implementados neste trabalho. Serão descritos: dois procedimentos para a contrução de uma solução inicial, onde um constrói uma solução inicial de forma gulosa e o outro através da inserção de pontos de Steiner na Árvore Geradora Mínima (AGM) construída para os pontos dados; um procedimento de busca local, onde foi utilizada a busca local em vetores topologia acrescida de modificações propostas nesta dissertação; um procedimento de perturbação e três procedimentos de aceitação, sendo utilizados os critérios de aceitar a melhor solução, aceitar uma solução cujo custo pertence a um intervalo obtido de um conjunto das melhores soluções encontradas e um critério que aceita aleatoriamente as soluções.

Também será descrita uma heurística, baseada em um modelo físico para o Problema de Steiner, denominada Relaxação Dinâmica Estendida. Iremos descrever os procedimentos chamados de evolução, responsável por determinar a posição dos pontos de Steiner, e de iteração, responsável pela modificação da estrutura da árvore. Será descrito como estes procedimentos são aplicados em conjunto à AGM, de forma a transformá-la em uma solução heurística para o PSE.

O ajuste dos parâmetros contidos nos algoritmos implementados, os resultados obtidos pela experiência computacional realizada neste trabalho e a respectiva análise são apresentados no capítulo 4. Por fim, no capítulo 5, apresentamos as conclusões e análises obtidas a partir de um conjunto de experimentos computacionais.

Capítulo 2

O Problema de Steiner

2.1 História

Podemos atribuir a origem do problema de Steiner a um problema proposto por Fermat, no início do século XVII, em seu artigo *Treatise on minima and maxima*. O problema consiste em encontrar o ponto, no plano, cuja soma de sua distância a outros três pontos dados seja mínima.

Segundo Kuhn[3], esse problema foi levado a Itália por Mersenne, que então encontrou nas mãos de Torricelli, em 1640, uma proposta geométrica de solução. O ponto encontrado por Torricelli pode ser obtido pela interseção de três circunferências, as quais circunscrevem três triângulos equiláteros, formados externamente a cada lado do triângulo construído com os três pontos dados como vértices. Este ponto é chamado de ponto de Torricelli ou de Fermat-Torricelli. Na figura 2.1 podemos ver tal ponto.

Uma propriedade interessante, com relação ao ponto de Torricelli, foi mostrada por Cavaliere em seu livro *Excercitationes Geometricae* de 1647. Esta propriedade consiste na igualdade dos três ângulos formados pelos segmentos de reta que ligam o ponto de Torricelli aos três pontos dados. Consequentemente, estes três ângulos são iguais a 120^o.

Em 1750, no livro *Doctrine and Application of Fluxions*, Simpson provou que os três segmentos de reta unindo os vértices externos de cada triângulo equilátero, na construção de Torricelli, aos respectivos vértices opostos do triângulo obtido pelos pontos dados se interceptam no ponto de Torricelli (ver figura 2.1). Hinen, em

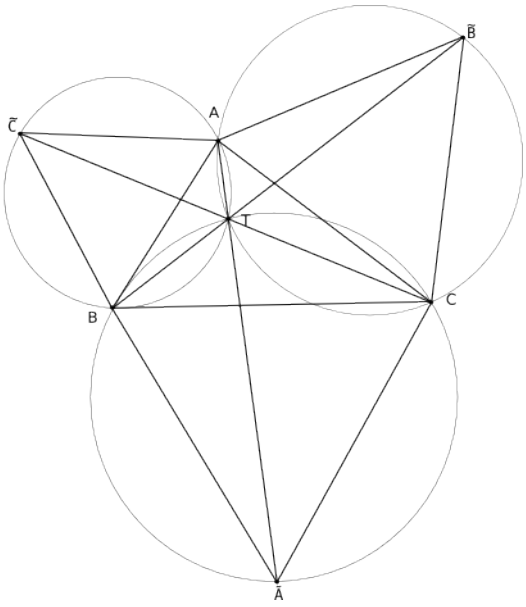

Figura 2.1: Construção do ponto de Torricelli para os pontos A, B e C

1834, provou que o tamanho dos segmentos de reta de Simpson são iguais entre si e também iguais à soma das distâncias entre o ponto de Torricelli e os três pontos dados, ou seja, seguindo a figura 2.1, para A, B e C pontos dados e T o ponto de Torricelli associado a eles, temos que

$$\overline{A\tilde{A}} = \overline{B\tilde{B}} = \overline{C\tilde{C}} = \overline{AT} + \overline{BT} + \overline{CT}. \tag{2.1}$$

No entanto, existem excessões onde a solução geométrica proposta por Torricelli não corresponde ao ponto que minimiza as distâncias. Na figura 2.2, podemos observar que o ponto de Torricelli não pertence mais ao interior do triângulo que tem como vértices os três pontos dados. Logo, não é o ponto que minimiza a soma das distâncias. Isso ocorre quando um dos ângulos do triângulo é maior ou igual a 120^o. Esse contra-exemplo da solução geométrica de Torricelli foi obtido por Hinen, em 1834, que reformulou a proposta de solução de Torricelli para uma solução completa do problema de Fermat.

Solução do problema de Fermat: *Se um dos ângulos do triângulo formado pelos pontos dados for maior ou igual a* 120^o*, o ponto que minimiza a distância é o vértice desse ângulo, caso contrário a solução é o ponto de Torriceli.*

O Problema de Fermat generalizado consiste em encontrar um ponto no plano,

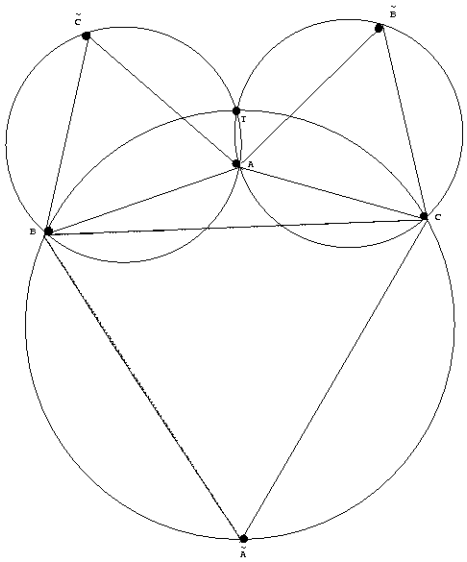

Figura 2.2: Contra-exemplo para a proposta de Torricelli de solução do problema de Fermat.

ou em um espaço n-dimensional, cuja soma das distâncias ou distâncias ponderadas deste ponto a p pontos dados seja mínima. Este problema é apresentado como um exercício no livro *Fluxions*, de Simpson, tendo se tornado muito popular no trabalho *Über den standart der industrien*, de Weber, relacionado à determinação da localização ótima de uma fábrica em relação aos fornecedores de matéria prima e aos consumidores. Por esse motivo o problema de Fermat é considerado o ponto de partida para a teoria da localização ótima[4].

Em uma carta para Schuhmacher, em 1836, Gauss propôs o problema de encontrar uma rede ferroviária que, com tamanho mínimo, conecte as cidades de Bremen, Harburg, Hannover e Braunschweig. A solução do problema é a rede que conecta Bremen, Harburg e Hannover a um ponto de Torricelli e que conecta Harburg diretamente a Braunschweig.

A questão levantada por Gauss é considerada um caso particular do problema de obter a rede de tamanho mínimo que interconecta p pontos no plano, a qual foi proposta por Jarnik e Kössler, em 1934, sendo por eles estudado o caso onde os pontos pertencem a um polígono regular com n lados.

No livro *What is mathematics?*, escrito por Courant e Robbins em 1941, o problema de encontrar a rede mínima que interconecta p pontos dados é denominado Problema de Steiner. Tal problema, nesse livro, não é relacionado ao problema de

Fermat, tampouco com o problema estudado por Jarnik e Kössler, mesmo o problema com p igual a 3 sendo o próprio problema de Fermat. Segundo Kuhn[3], apesar de Steiner ter trabalhado no problema, nenhuma contribuição ao tema foi atribuída a ele para que o mesmo fosse batizado com seu nome. Gieslick[4] também aponta como sendo um grande erro não mencionar o problema de Fermat, nem Gauss ou Jarnik e Kössler, atribuindo a fixação do nome ao problema tão somente à popularidade do livro.

2.2 Definições e Propriedades

Existem diferentes versões para o Problema de Steiner, todas procurando obter a rede de tamanho mínimo que interconecta p pontos dados. Porém estas se diferenciam pelo conjunto ao qual os pontos dados pertencem, por exemplo $\Re^2$, e o tipo de rede a ser procurada, que pode ser um subgrafo de um grafo dado ou uma rede cujas ligações perfazem angulos de 90^0, entre outras. Dentre todas as versões, podemos citar o Problema de Steiner em Grafos[5], o qual consiste em dado um grafo, direcionado ou não, e uma função custo associada as suas arestas, encontrar um subgrafo, acíclico e conexo, que conecta um subconjunto dos vértices deste grafo, tal que a soma dos custos das arestas deste subgrafo seja mínima. Outro exemplo é o Problema de Steiner Rectilíneo[6], o qual tem como objetivo encontrar a menor rede que interconecta o conjunto de pontos dados utilizando apenas ligações horizontais ou verticais.

O presente trabalho de dissertação tratará da resolução do problema de Steiner no espaço Euclidiano n dimensional. Neste problema, o objetivo é encontrar a rede de tamanho mínimo que interconecta p pontos dados no $\Re^n$,onde a distância entre os pontos é calculada utilizando-se a norma euclidiana,

$$\|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}. \tag{2.2}$$

Como se objetiva encontrar uma rede conexa que minimiza uma soma de distâncias, tal rede não pode conter ciclos, pois a remoção de qualquer uma das conexões de um ciclo resultará em uma rede menor sem desconectá-la. Como a rede não deve possuir ciclos e deve ser conexa, podemos restringir o problema a encontrar árvores

com tamanho mínimo, definindo o problema de Steiner da seguinte forma:

Problema de Steiner Euclidiano em $\Re^n$ (PSE):

Dado um conjunto finito de pontos no espaço n-dimensional, deve-se encontrar a menor árvore que os interconecta, podendo ou não serem adicionados pontos extras.

Os pontos extras adicionados na árvore, além dos p pontos dados, são chamados de pontos de Steiner. Neste trabalho utilizaremos algumas nomenclaturas usadas em teoria dos grafos, podendo os pontos dados e os pontos de Steiner serem chamados de vértices e as ligações entre os pontos da árvore de arestas.

Algumas propriedades referentes à árvore obtida como solução do problema foram demonstradas por Gilbert e Pollak[7] e serão apresentadas no decorrer desta seção.

Dada uma árvore, consideramos dois tipos de operações que alteram sua quantidade de vértices: as operações de encolhimento de uma aresta e de divisão de um vértice. O encolhimento de uma aresta como a operação que a remove da árvore, e consequentemente, torna os seus vértices adjacentes em um único vértice. Por outro lado, a operação de divisão de um vértice v consiste em criar um novo vértice v'(ponto de Steiner), onde as arestas adjacentes a v tornam-se adjacentes a v'. Em seguida, uma aresta conectando os vértices v e v' é criada. Note que estas operações são inversas uma da outra , ou seja, uma operação desfaz a outra.

Uma árvore cujo comprimento não pode ser reduzido pela alteração da posição de seus pontos de Steiner e nem pelas operações de encolhimento de aresta ou divisão de vértice é chamada de árvore de Steiner. A menor dentre todas as árvores de Steiner para um conjunto de pontos dado é a árvore mínima de Steiner, a qual é a solução do Problema de Steiner Euclidiano.

Um ponto de Steiner não pode ter grau um, pois este seria uma folha da árvore e sua eliminação resultaria em uma árvore de comprimento menor. Também não é possível obter um ponto de Steiner com grau dois, já que pela desigualdade triangular podemos obter uma árvore menor através da remoção desse ponto de Steiner e a criação de uma aresta para ligar os seus pontos adjacentes. Com isso, um ponto de

Steiner em uma árvore mínima de Steiner tem que ter grau maior ou igual a três.

Considerando um ponto da árvore, suponhamos que este tenha um ângulo menor que 120^0 com relação a duas de suas arestas adjacentes. Então, a adição de um ponto de Steiner exatamente na posição do ponto de Torricelli resultaria em uma árvore de comprimento menor. Logo, nenhum ponto deve ter um ângulo menor que 120^0 entre duas arestas adjacentes, em uma árvore mínima de Steiner.

Dado um ponto de Steiner em uma árvore mínima de Steiner, se o ângulo entre um par de arestas adjacentes for maior que 120^0 e o número de ligações deste ponto com outros pontos da árvore for maior que três, temos que o ângulo entre pelo menos outro par de arestas adjacentes será menor que 120^0, não podendo esta árvore ser mínima. Com isso, o ângulo entre um par de arestas adjacentes a um ponto de Steiner deve ser exatamente 120^0, e consequentemente, o seu grau deve ser igual a 3.

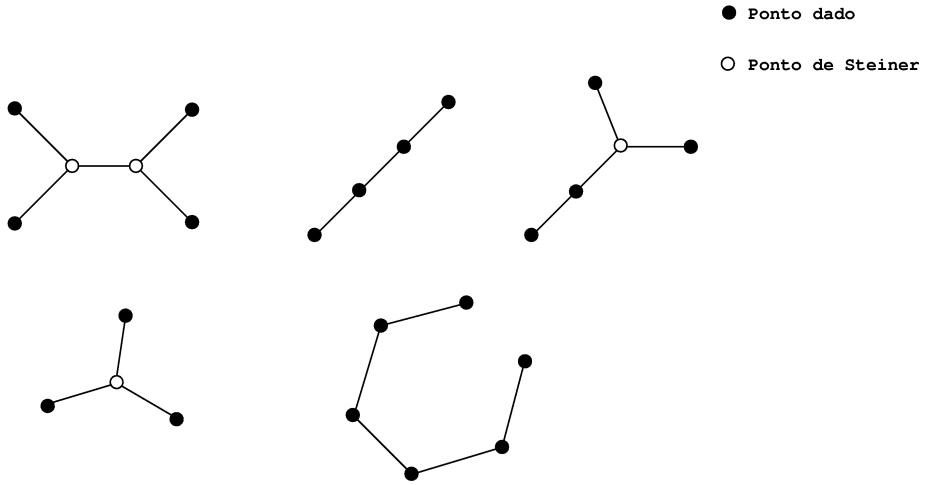

Figura 2.3: Exemplos de árvores de Steiner

Dada uma árvore com p pontos dados e s pontos de Steiner, temos que esta tem $p + s - 1$ arestas. Além disso, como cada ponto de Steiner possui três arestas adjacentes e cada ponto dado possui no mínimo uma aresta, então, no mínimo, existem $\frac{p+3s}{2}$ arestas , já que cada aresta é contada uma vez para cada vértice. A partir destas observações, chegamos nas desigualdades abaixo:

$$p + s - 1 \geq \frac{3s + p}{2} \Rightarrow 2p + 2s - 2 \geq 3s + p \Rightarrow p - 2 \geq s. \tag{2.3}$$

Dessa forma, podemos concluir que o número máximo de pontos de Steiner é $p - 2$.

O conjunto das ligações entre os pontos dados e os pontos de Steiner de uma árvore é chamada de topologia. Caso todos os pontos de Steiner dessa árvore tenham grau três, então dizemos que ela possui uma topologia de Steiner. Se uma topologia de Steiner contém exatamente $p-2$ pontos de Steiner, esta é chamada de topologia cheia.

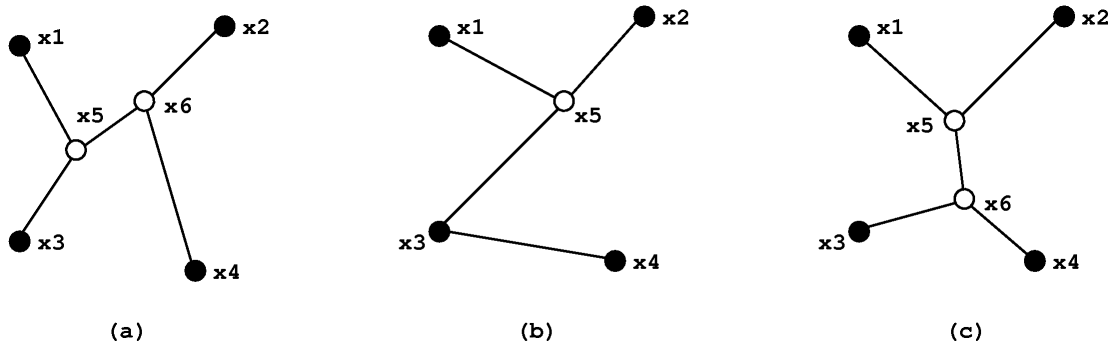

Figura 2.4: Três topologias de Steiner com os mesmos quatro pontos dados, sendo as topologias a e c cheias.

Para a resolução do Problema de Steiner Euclidiano, dois problemas devem ser simultaneamente considerados. O primeiro consiste em encontrar a posição dos pontos de Steiner relativos a uma topologia, tal que a soma do tamanho de suas ligações seja mínima, sendo a árvore obtida uma árvore mínima relativa à topologia de Steiner dada. O outro problema consiste em encontrar a topologia que, ao ser minimizada, resulte na árvore de Steiner de menor tamanho dentre todas as outras árvores de Steiner possíveis para o conjunto de pontos dados. Com a divisão do PSE nestes dois problemas, podemos observar que cada posicionamento ótimo dos pontos de Steiner para uma dada topologia é um mínimo local. Além disso, o mínimo global só é atingido com a topologia que permite um posicionamento tal que a soma do tamanho de suas ligações seja menor dentre todas as topologias possíveis para o conjunto de pontos dados.

Seja $\|.\|$ a distância euclidiana 2.2. Temos que, para as topologias apresentadas na figura 2.4, e considerando o problema de Steiner no plano, os problemas de posicionamento ótimo são:

a) $MIN\ \|x_1 - x_5\| + \|x_3 - x_5\| + \|x_2 - x_6\| + \|x_4 - x_6\| + \|x_5 - x_6\|$

Sujeito a $x_5 \in \Re^2$ e $x_6 \in \Re^2$

b) $MIN\ \|x_1 - x_5\| + \|x_2 - x_5\| + \|x_3 - x_5\| + \|x_3 - x_4\|$

Sujeito a $x_5 \in \Re^2$

c) $MIN\ \|x_1 - x_5\| + \|x_2 - x_5\| + \|x_3 - x_6\| + \|x_4 - x_6\| + \|x_5 - x_6\|$

Sujeito a $x_5 \in \Re^2$ e $x_6 \in \Re^2$

A solução para cada um dos problemas acima é dada na figura 2.5.

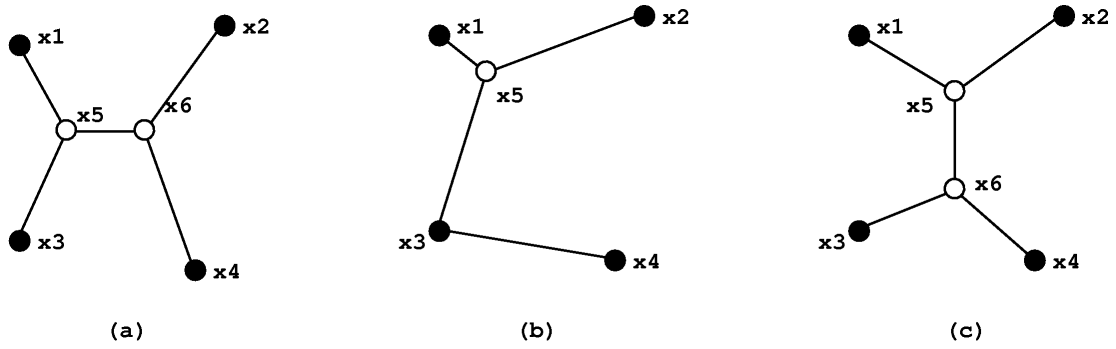

Figura 2.5: Árvores mínimas relativas

Nas topologias a e c obtemos como resultado uma árvore de Steiner. Já na b obtemos uma árvore mínima relativa, que não é uma árvore de Steiner pois o ângulo entre as arestas $(x5, x3)$ e $(x4, x3)$ é menor que 120^0. Neste caso, uma operação de separação resultaria em uma árvore menor.

Com isso verificamos que a minimização do tamanho de cada ligação de uma árvore com topologia de Steiner não garante que obteremos como resultado uma árvore de Steiner.

De forma a evitar a minimização de topologias que não resultarão em árvores de Steiner, introduziremos o conceito de topologia degenerada[8].

Uma topologia B é considerada degenerada de uma topologia A se B pode ser obtida a partir de A pela operação de encolher uma aresta, ou seja, se a topologia B for igual a topologia A após considerarmos uma ou mais arestas (ligando um ponto dado à um ponto de Steiner ou ligando dois pontos de Steiner) como tendo

tamanho igual a zero. Dessa forma, tomando D(A) como o conjunto de todas as topologias degeneradas de A, onde B ∈ D(A), podemos considerar que toda topologia de Steiner, T_s, pode ser obtida a partir de uma topologia de Steiner cheia, T_F, tal que $T_s \in D(T_F)$, não sendo esta topologia de Steiner cheia necessariamente única.

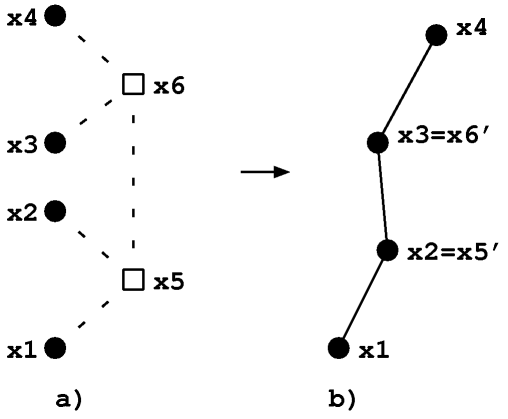

Figura 2.6: Topologia de Steiner cuja árvore mínima relativa é degenerada entre pontos dados e pontos de Steiner.

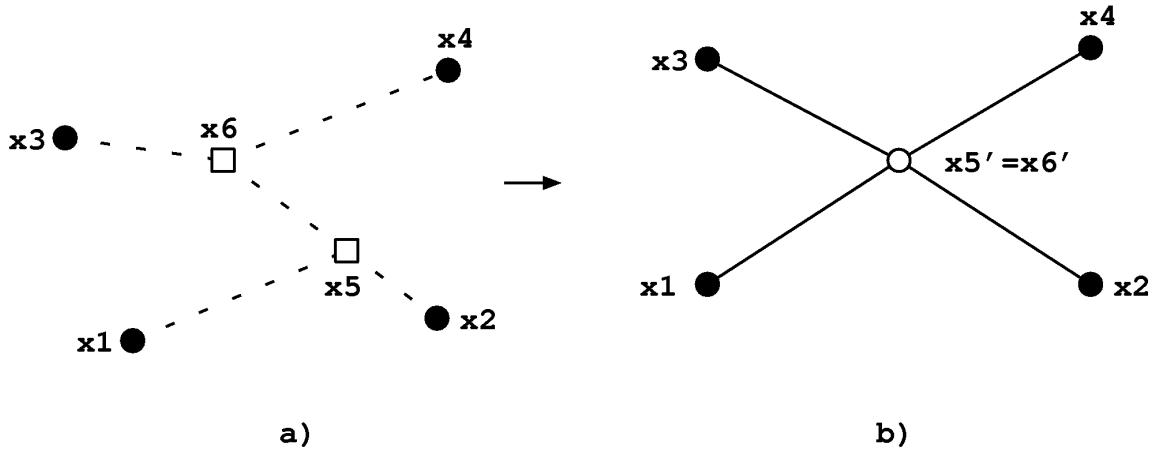

Figura 2.7: Topologia de Steiner cuja árvore mínima relativa é degenerada entre dois pontos de Steiner.

O seguinte teorema garante que, ao minimizarmos uma topologia de Steiner cheia (T_F), a árvore obtida será a menor dentre todas as árvores pertencentes a $D(T_F)$.

Teorema 1: Uma árvore de Steiner cuja topologia pertence a $D(T_F)$ para alguma topologia de Steiner cheia T_F é a única árvore de Steiner dentre as árvores mínimas cuja topologia de Steiner pertence a $D(T_F)$.

Demonstração. Seja A uma árvore cuja topologia $T \in D(T_F)$ e, para as aresta $(i,j) \in T_F$, a seguinte relação:

$$\begin{matrix} f_{ij} = 1, \text{ se } (i,j) \in T \\ f_{ij} = 0, \text{ caso contrário} \end{matrix}, \tag{2.4}$$

onde f_{ij} é responsável por sinalizar a presença de uma aresta na topologia T. Temos que o tamanho da árvore com topologia T é igual a

$$|A| = \sum_{i<j} f_{ij} \|x_i - x_j\|. \tag{2.5}$$

Suponha que B é a árvore de menor tamanho com topologia T_F. Como a função modular é estritamente convexa e a soma de funções convexas é uma função convexa, temos que $|B|$ é uma função convexa. Esta função não é estritamente convexa se para algum ponto de Steiner uma alteração na sua posição não resultar em uma modificação no valor de $|B|$. No entanto, essa modificação mudaria o valor dos ângulos entre as arestas adjacentes à esse ponto de Steiner, com isso não sendo mais iguais a 120^0, e consequentemente, a árvore não sendo mínima. Dessa forma, como a árvore B é a única árvore com tamanho $|B|$ e nenhum ponto de Steiner pode ser inserido à sua topologia, temos que esta é uma árvore de Steiner.

Se alterarmos a função $|B|$ de maneira que para cada aresta, (i,j), com tamanho nulo o valor f_{ij} seja igual a 0, então a função encontrada $|A|$ está associada a uma árvore A cuja topologia pertence a $D(T_F)$. Como $|A| = |B|$, então A é uma árvore de Steiner com topologia $T \in D(T_F)$.

No entanto, se alterarmos os valores de f_{ij} na função $|B|$ para que esta seja igual a função associada a uma outra árvore com topologia pertencente a $D(T_F)$, teremos, após a sua minimização que, o tamanho dessa árvore será maior que $|B|$, e uma mudança em sua topologia resultaria em uma árvore de menor tamanho. Com isso, essa árvore não pode ser uma árvore de Steiner. Logo, podemos concluir que A é a uníca árvore de Steiner com topologia pertencente a $D(T_F)$. □

Portanto, para encontrar todas as árvores mínimas relativas para um conjunto de pontos dados, é suficiente minimizar todas as topologias de Steiner cheias possíveis para o conjunto.

Através do teorema anterior podemos garantir que, ao minimizarmos as topologias de Steiner cheias para o conjunto de pontos dados, obteremos árvores mínimas

relativas que não terão o seu tamanho reduzido pela inserção de um novo ponto de Steiner e nem pela perturbação da localização dos pontos de Steiner já existentes.

No entanto, mesmo considerando apenas o conjunto de topologias de Steiner cheias, ainda nos deparamos com árvores mínimas relativas, obtidas pela minimização de árvores com topologia de Steiner cheia, que não são árvores de Steiner.

Esta situação ocorre quando a topologia da árvore mínima relativa é degenerada de tal forma que uma ou mais ligações entre dois pontos de Steiner são iguais a zero, o que é equivalente a obter uma topologia que possui pontos de Steiner com grau quatro.

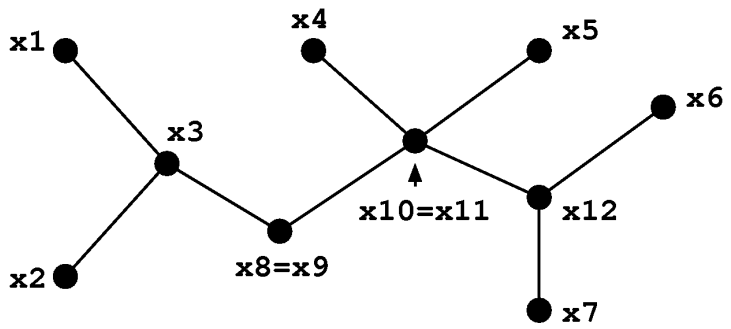

Figura 2.8: Árvore mínima relativa com topologia degenerada.

Porém, ao considerar estes dois pontos de Steiner sobrepostos como um único, teremos um ponto de Steiner com grau quatro, e consequentemente, o ângulo entre suas ligações será menor que 120^0. Com isso, podemos encontrar uma árvore com tamanho menor ou igual ao da árvore anterior pela inserção de um ponto de Steiner. Tal ponto será conectado a duas das arestas ligadas ao ponto de Steiner com grau quatro e a uma nova aresta criada para ligar este novo ponto de Steiner ao ponto de Steiner que tinha quatro ligações.

Mesmo com a função convexa 2.2 (considerada na minimização) não sendo diferenciável, podemos encontrar o seu mínimo usando alguns métodos específicos[8].

Em duas dimensões, a minimização de uma árvore com uma topologia não-degenerada pode ser realizada utilizando um algoritmo de complexidade linear proposto por Hwang e Melzak[8]. O algoritmo divide a árvore em subárvores com topologia de Steiner cheia. Em seguida, cada uma das árvores é minimizada. A minimização das subárvores com 3 e 4 pontos dados é realizada de forma direta. Já as subárvores contendo mais de 5 pontos dados são minimizadas utilizando um

procedimento que reduz a subárvore à uma árvore com 4 pontos dados que, após ser minimizada, a subárvore é reconstruída através da inserção dos pontos dados, que foram retirados, e seus pontos de Steiner adjacentes são posicionados de maneira ótima. Após minimizar todas as subárvores, estas são concatenadas para formar a árvore relativa mínima. A baixa complexidade deste algoritmo é justificada pelo fato de que o posicionamento de qualquer ponto pode ser encontrado a partir de outro, já que os ângulos entre as suas arestas adjacentes têm que ser, na posição mínima, exatamente 120^0. Esta facilidade, que não é encontrada para dimensões maiores que dois[9], é uma particularidade para pontos pertencentes ao plano.

Para dimensões maiores que dois é utilizado um método iterativo, desenvolvido por Smith [9], para obter a posição ótima dos pontos de Steiner para uma dada topologia de Steiner cheia.

A simples minimização de uma topologia de Steiner cheia produz um mínimo local para o problema, sendo necessário encontrar a topologia de Steiner cheia que, ao ser minimizada, resulte na árvore de Steiner de menor tamanho, ou seja, a árvore mínima de Steiner. Em função dessas considerações, apresentamos a seguir a fórmula associada ao número de topologias de Steiner que podem ser construídas, em particular, o número de topologias de Steiner cheias.

Considerando o número de pontos dados como p e como s o número de pontos de Steiner pertencentes à topologia, o número de topologias de Steiner possíveis $F(p, s)$[7] é igual a

$$F(p,s) = \begin{pmatrix} p \\ s+2 \end{pmatrix} \frac{(p+s-2)!}{s!2^s} \ . \tag{2.6}$$

Se nos restringirmos somente as topologias cheias, temos que $s = p - 2$ e

$$F(p, p-2) = \frac{(2p-4)!}{(p-2)!2^{p-2}} \ . \tag{2.7}$$

Ao utilizarmos, tal como feito em [10], a identidade

$$(2m+1)! = 2^m m!(2m+1)!! \ , \tag{2.8}$$

onde $(.)!!$ denota o duplo fatorial, a expressão $F(p, p-2)$ se reduz para

$$F(p) = (2p-5)!! \ . \tag{2.9}$$

O crescimento super-exponecial do número de topologias cheias[10] torna proibitiva a aplicação de um algoritmo do tipo força bruta para obter a topologia associada à árvore mínima de Steiner. Isso pode ser observado mesmo para uma pequena quantidade de pontos, pois $p = 10$, por exemplo, temos que o número de topologias cheias possíveis é de $15!! = 1.3.5.7.9.11.13.15 = 2.027.025$.

A formulação do Problema de Steiner Euclidiano como um problema de decisão mostra que este pertence à classe dos problemas NP-Difícil, e ao discretizarmos a norma euclidiana o mesmo pertence a classe dos problemas NP-completo[2]. Dessa forma, considerando que $P \neq NP$, este problema é tão complicado quanto qualquer outro problema da classe NP-completo, os quais não possuem um algoritmo determinístico para encontrar soluções exatas em tempo polinomial[11].

2.3 Formulações

Entre os diversos modelos propostos na literatura para o Problema de Steiner Euclidiano, apenas dois correspondem a modelos de programação matemática inteiro misto e foi proposto por Maculan et al. [12] e Fampa & Maculan [13]. Os demais modelos são baseados em sistemas físicos e são importantes para obtenção de métodos que minimizam as topologias de Steiner de forma eficiente.

O modelo de programação matemática proposto por Maculan tem as propriedades de não-convexidade e não-diferenciabilidade no problema após a sua relaxação. Além disso, considera como entrada um grafo $G = (V, E)$, tal que $V = P \cup S$, onde $P = \{1, 2, ..., p-1, p\}$ é o conjunto dos índices associados a cada ponto dado, $S = \{p+1, p+2, ..., 2p-3, 2p-2\}$ o conjunto dos índices associados a cada ponto de Steiner e E o conjunto das arestas $[i, j]$ adjacentes aos vértices

i, $j \in V$. O modelo consiste em:

$$\text{minimizar } \sum_{[i,j]\in E} \|x^i - x^j\| y_{ij} \tag{2.10}$$

$$\text{s.a.: } x_i \in \Re^n,\ i \in \{p+1, p+2, ..., 2p-2\}, \tag{2.11}$$

$$\sum_{j\in S} y_{ij} = 1,\ i \in P \tag{2.12}$$

$$\sum_{i\in P} y_{ij} + \sum_{k\leq j,\ k\in S} y_{kj} + \sum_{k\geq j,\ k\in S} y_{jk} = 3,\ j \in S \tag{2.13}$$

$$\sum_{k\leq j,\ k\in S} y_{kj} = 1,\ j \in S - \{p+1\}, \tag{2.14}$$

$$y_{ij} \in \{0, 1\},\ [i, j] \in E \tag{2.15}$$

Nesse modelo, y_{ij} é uma variável binária que assume valor 1 se a aresta $[i, j] \in E$ pertence à árvore mínima de Steiner. Caso contrário, $y_{ij} = 0$.

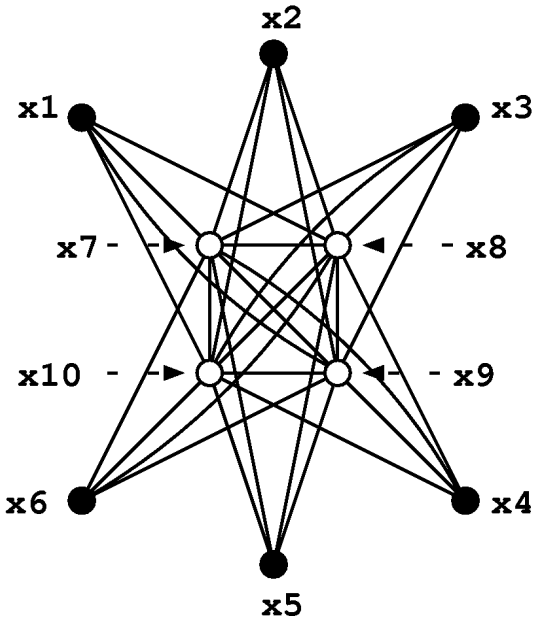

Figura 2.9: O grafo $G = (V, E)$ considerado na formulação matemática do PSE, onde o subgrafo induzido pelos pontos de Steiner é uma clique.

A restrição 2.12 mantém o grau de cada vértice em P igual a um. Por sua vez, a restrição 2.13 fixa o grau dos vértices pertencentes a S em 3. Como uma árvore não pode conter ciclos, a restrição 2.14 impede a formação de ciclos nas soluções viáveis. Desta forma, o conjunto de soluções viáveis que satisfazem as restrições 2.12-2.15 é igual ao conjunto de todas as topologias cheias possíveis para o conjunto de pontos dados.

Nesse mesmo trabalho, além da formulação matemática, Maculan et al. propuseram uma relaxação lagrangiana para o problema objetivando-se encontrar lim-

itantes duais (bounds), através da utilização do método do subgradiente, para o PSE. Tais limites podem ser utilizados em conjunto com o algoritmo de enumeração implícita proposto por Smith[9] para encontrar soluções exatas para o PSE em $\Re^n$ de maneira mais eficiente.

Um outro paradigma para o PSE consiste na interpretação da topologia de Steiner como um sistema mecânico, cuja energia potencial é a soma do tamanho de todas as arestas da topologia de Steiner. Quando este sistema entra em equilíbrio, ou seja, quando tiver energia potencial mínima, a árvore encontrada associada à topologia de Steiner (sistema) será uma árvore de Steiner.

Baseado no modelo apresentado por Gilbert et al.[7], Smith[9] modelou uma topologia de Steiner como um sistema de molas ideais obedecendo a lei de Hooke. Estas molas substituem cada aresta da topologia de Steiner, sendo a constante elástica de uma mola inversamente proporcional à distância entre o ponto de Steiner e o ponto dado conectados por ela. Dessa forma, ao minimizarmos a energia potencial desse sistema de molas, encontraremos a árvore mínima relativa relacionada com a topologia de Steiner.

Gilbert et al.[7] demostraram todas as propriedades referentes às árvores de Steiner mínimas através desse modelo. Além desse, existem outros modelos baseados em interpretações físicas do problema.

Em [14] e [15] é apresentado um modelo físico para o PSE construído através de uma analogia entre uma película de sabão e uma árvore de Steiner. Essa película de sabão é formada pela imersão de duas placas presas paralelamente uma a outra por bastões posicionados nos pontos dados em uma solução com sabão. Após esse dispositivo ser retirado da solução, a película formada sofre tensões de forças superficiais e relaxa até assumir uma forma tal que as forças estejam em equilíbrio. A forma dessa película em equilíbrio apresenta uma configuração com energia mínima e padrão semelhante a uma árvore de Steiner para os pontos dados. Na figura 2.10 é ilustrado o padrão obtido pelo experimento.

Como no processo de relaxação a película modifica a sua topologia procurando atingir o equilíbrio, observamos que a árvore relacionada com a configuração de energia mínima da película tende à árvore mínima de Steiner. Para mais detalhes ver [14].

Com base nestas características, este modelo é extremamente atraente para a construção de métodos para encontrar a solução para o PSE. Em uma seção seguinte veremos com mais detalhes a heurística de Relaxação Dinâmica[16]. Esta heurística tenta encontrar uma solução de boa qualidade para o PSE através de um procedimento que tenta simular a dinâmica de uma película de sabão.

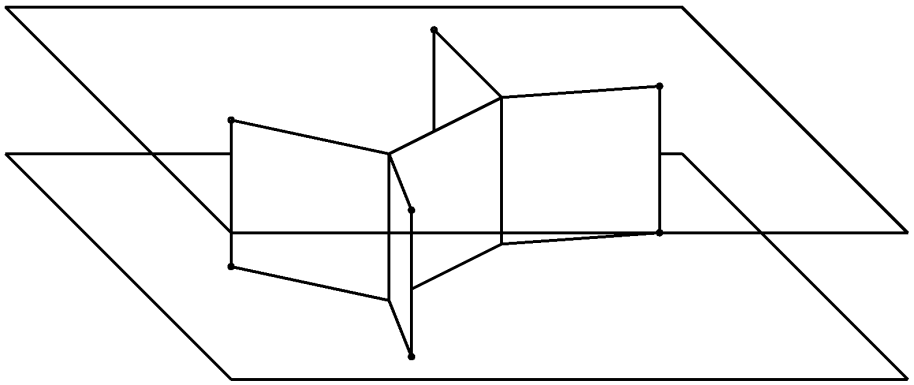

Figura 2.10: Película de sabão

Outros modelos para o PSE são apresentados no trabalho de Soukup[17].

2.4 Aplicações

Com relação à aplicabilidade do Problema de Steiner Euclidiano, podemos utilizá-lo como modelo para qualquer problema onde o objetivo é encontrar uma rede que liga determinados pontos, objetos ou localidades, com tamanho mínimo. Observando a origem histórica do PSE, antes mesmo de receber esse nome, este foi proposto por Gauss, tal como vimos acima, como um problema de construir uma rede ferroviária de tamanho mínimo para conectar um conjunto de cidades alemãs.

Sua aplicação na definição de redes é extremamente importante do ponto de vista econômico, uma vez que se tivermos um custo associado à rede proporcional ao seu tamanho, a rede de menor custo possível será a Árvore Mínima de Steiner.

As redes podem ser bidimensionais, como no caso da construção de rede de dutos de ventilação[18], e tridimencionais, como no problema da construção de uma rede de túneis para mineração [18] [19].

Dado um conjunto qualquer de pontos pertencentes a um espaço Euclidiano n-dimensional, se considerarmos forças unitárias para cada ponto deste conjunto,

temos que pelo teorema de Maxwell[7] a uma árvore mínima relativa com topologia de Steiner cheia apresenta uma configuração estável. A posição dos pontos de Steiner faz com que as forças associadas a cada ponto dado entrem em equilíbrio. Dessa forma, podemos ver que o problema de encontrar a Árvore Mínima de Steiner é equivalente ao problema de encontrar uma configuração estável para os pontos que minimize a energia potencial entre eles.

Através dessa equivalência, podemos utilizar o problema de Steiner para solucionar qualquer problema onde o objetivo é encontrar uma configuração de energia mínima para um conjunto de pontos cujas as forças são uniformes. Se essas não forem uniformes, a Árvore Mínima de Steiner corresponde a apenas um limitante inferior para o problema.

Frequentemente, nos deparamos com estruturas que apresentam uma configuração de energia mínima, sendo o PSE um alternativa para encontrar limitantes para estas estruturas, quando desconhecidas, e em casos particulares sendo possível encontrá-las.

Tentando explorar esta relação, McGregor Smith[20] comparou a estrutura das Árvores Mínimas de Steiner para conjuntos de pontos situados no espaço tridimensional com estruturas moleculares espaciais de proteínas. Ele verificou que a soma de todas as distâncias interatômicas era aproximadamente igual ao tamanho da Árvore Mínima de Steiner quando os pontos dados são colocados exatamente na posição dos átomos da proteína.

Dessa maneira, encontrou árvores mínimas de Steiner relacionadas a algumas configurações helicoidais de baixa energia potencial, similares a estruturas de macromoléculas orgânicas como o DNA e o RNA[10]. Também foram conduzidos por McGregor Smith[21] experimentos para verificar a viabilidade na aplicação do PSE para encontrar a estrutura secundária de proteínas.

Dados N elementos, cada um com m características, a árvore que conecta todos esses elementos com base na diferença entre suas características é chamada de árvore filogenética, ou árvore evolutiva. Onde cada um dos elementos representa uma espécie e suas caracteristicas podendo ser morfológicas ou moleculares, como sequências de DNA ou proteínas comuns a todas as espécies consideradas. Existem diversas metologias dedicadas a encontrar árvores filogenéticas. Dentre elas,

o Método da Evolução Mínima[22] obtém uma solução aproximada encontrando a rede de menor tamanho que conecta as espécies através de suas características em um espaço euclidiano. Dessa forma, o Problema de Steiner Euclidiano pode ser utilizado para encontrar tais redes de tamanho mínimo, e consequentemente, a árvore filogética para um conjunto de espécies. Como os pontos que serão utilizados na árvore pertencem ao espaço euclidiano de dimensão igual ao número de características, então é utilizado o PSE em $\Re^m$.

Uma outra aplicação, além de encontrar árvores filogenética, para o PSE em $\Re^n$ e a de encontrar agrupamentos em conjuntos de dados[10].

2.5 Razão de Steiner

Antes de comparar os diferentes métodos heurísticos utilizados para resolver o Problema de Steiner Euclidiano, faz-se necessário apresentar uma medida de qualidade para a solução obtida por métodos heurísticos. Para tal, utiliza-se uma medida denominada a razão de Steiner, apresentada na equação abaixo:

$$\rho = \frac{L(AS)}{L(AGM)}, \tag{2.16}$$

onde $L(.)$ denota o comprimento de uma árvore, AS é uma árvore de Steiner e AGM é a árvore geradora mínima para os pontos dados.

Frequentemente, esta razão é utilizada para avaliar as soluções encontradas por métodos heurísticos, onde substituímos na equação 2.16 a árvore de Steiner (AS) por uma árvore mínima de Steiner (AMS_H) encontrada por um método heurístico. Considerando ρ^* a razão de Steiner relativa à árvore mínima de Steiner obtida por um algoritmo exato e ρ_h a razão obtida por um algoritmo heurístico, temos que $\rho* \leq \rho_h$. Dessa forma, ρ_h é um limitante superior para a razão de Steiner ótima.

Consideramos que uma solução aproximada é de boa qualidade se $\rho_h \leq 1$ é o mais próxima possível de ρ^*. Conforme exposto anteriormente, o calculo de ρ^* é proibitivo para instâncias contendo mais de 20 pontos. Devido a esta restrição em [10] é colocado que a proximidade de uma solução ótima pode ser medida por quão próximo $\Delta\rho = \rho_d - \rho_h$ está de zero, onde ρ_d é a razão de Steiner para uma solução dual do problema de Steiner, tal que $\rho_d \leq \rho^*$, sendo ρ_d portanto um limitante inferior para a razão de Steiner ótima.

Considerando todas as distribuições de pontos dados possíveis para um espaço euclidiano n dimensional, um grande esforço tem sido realizado para obter um limitante inferior para o menor valor que a razão de Steiner pode assumir para estas distribuições. Esse limitante (ρ_n) é encontrado quando

$$\rho_n = inf_{AMS \in \Re^n} \ \rho(AMS). \tag{2.17}$$

No entanto, estes limitantes, por serem muito gerais, são extremamente difíceis de serem encontrados. No plano foi demonstrado por Du e Hwang[23]que a conjectura apresentada por Gilbert e Pollak em [7] de que $\rho_2 = \frac{\sqrt{3}}{2}$ é realmente o menor valor que ρ^* pode assumir para um PSE qualquer, cujo os pontos pertensem ao plano. No entanto, esta conjectura foi provada não ser verdadeira para dimensões maiores que dois [24]. Para dimensões $n \geq 2$ temos apenas que $\rho_n \geq 0.615827$.

Smith et al.[1], através de uma estrutura denominada *sausage*, ver figura 2.11, conjecturaram que o ínfimo do valor de ρ para o $\Re^3$ é aproximadamente $0,784190373$. Este foi o melhor limitante presente na literatura durante alguns anos, até que configurações tridimensionais com razões de Steiner ainda menores, relacionadas também a estruturas de biomoléculas, foram reportadas [25].

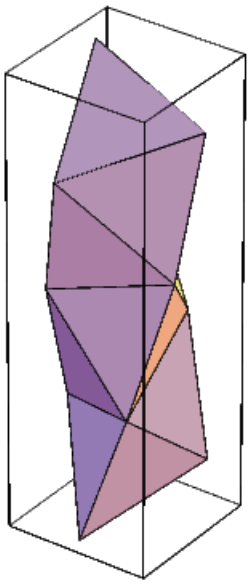

Figura 2.11: Uma *sausage* para 11 pontos em $\Re^3$[1]

2.6 Revisão Bibliográfica

Nas seções anteriores apresentamos propriedades relacionadas com as árvores mínimas de Steiner e também expomos a alta complexidade computacional para encontrá-las devido a complexidade exponencial deste problema.

No entanto, no caso $\Re^2$, diversos métodos exatos e heurísticos eficientes foram desenvolvidos, produzindo soluções ótimas ou muito próximas das ótimas. Maiores detalhes sobre estes métodos podem ser encontrados em [8].

O sucesso desses métodos para resolver o Problema de Steiner Euclidiano no plano se deve a algumas propriedades geométricas do problema no plano. Para este particular caso, podemos destacar a não interseção de arestas em uma árvore de Steiner, a divisão de uma topologia de Steiner em subtopologias de Steiner cheias e a presença de algoritmos lineares para encontrar a árvore mínima relativa relacionada a uma árvore com uma topologia de Steiner cheia.

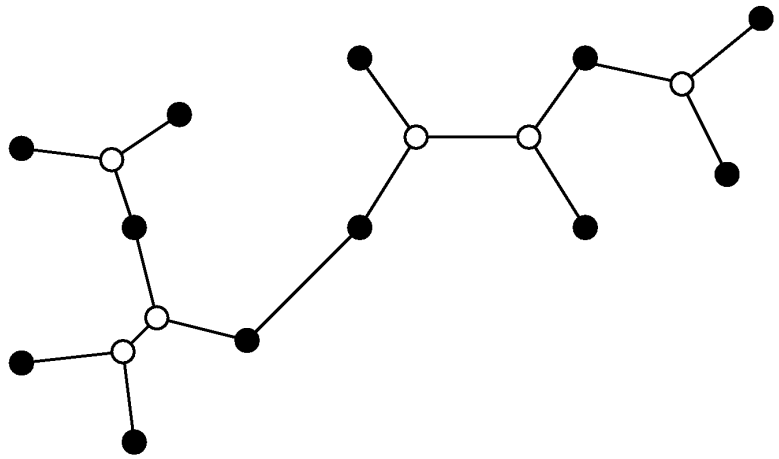

Figura 2.12: Topologia de Steiner contendo sub-topologias de Steiner cheias

Com isso, algoritmos que dividem uma árvore em árvores menores com topologia cheia, em seguida minimizam estas árvores e depois as reconcatenam tornam-se viáveis. Como também, encontram soluções com valores muito próximos dos ótimos utilizando pouco tempo de CPU.

Já em dimensões maiores que 2, as propriedades geométricas do plano não podem ser aplicadas uma vez que estas não restringem o conjunto de soluções possíveis, pelo fato de, nessas dimensões mais altas, normalmente as árvores mínimas de Steiner não apresentarem subtopologias cheias e de raramente duas arestas se cruzarem.

Atualmente existem dois algoritmos exatos para solucionar o problema de Steiner Euclidiano em $\Re^n$, sendo um deles uma versão melhorada do outro.

O algoritmo exato desenvolvido por Smith[9] utiliza uma estrutura chamada vetor topologia, a qual representa de maneira única cada topologia de Steiner cheia possível para uma distribuição de pontos dados utilizando um vetor de $p-3$ posições.

Com esta estrutura, o algoritmo é capaz de enumerar cada uma das soluções em um procedimento de enumeração implícita similar a um algoritmo *branch-and-bound*. A enumeração é realizada através da construção de um vetor topologia, onde cada nó da árvore de enumeração corresponde a um vetor topologia para um subconjunto dos pontos dados. Inicialmente, este conjunto possui apenas três pontos dados e a cada nível da árvore de enumeração um ponto dado é inserido no conjunto.

Para avaliar o tamanho da árvore relativa mínima associada a um vetor topologia, a árvore referente ao vetor topologia é minimizada através de um método iterativo baseado em uma analogia entre um sistema de molas ideais em equilíbrio e uma árvore mínima relativa.

Esse algoritmo será descrito detalhadamente na seção seguinte, tendo em vista que o mesmo serviu de base para a implementação das propostas metodológicas desta dissertação.

Fampa et al.[26] introduziram uma melhoria no algoritmo exato de Smith. Nesta melhoria, em vez de minimizar as árvores referentes aos vetores topologia, realizando iterativamente a solução de sistemas lineares, optou-se pela resolução de um modelo de programação matemática [12], mediante a aplicação de uma relaxação cônica [13]. Utilizando este modelo, há a possibilidade de verificar quais pontos são mais influentes na minimização da árvore, fazendo um *strong-branch* [27] na árvore de enumeração de topologias de Steiner cheias.

Mesmo ampliando a resolução exata do PSE em $\Re^n$ para instâncias contendo uma quantidade maior de pontos dados, essa proposta ainda continua proibitiva para instâncias com distribuição contendo mais de 20 pontos dados[18].

Diferentes heurísticas foram propostas para solucionar o problema em dimensões $n \geq 3$, sendo algumas específicas para distribuições de pontos em $\Re^3$ enquanto que outras, geralmente mais recentes, procuram também soluções em $n > 3$.

As heurísticas que lidam apenas com conjuntos de pontos em $\Re^3$ são tentativas

de utilizar a técnica de dividir para conquistar do plano no espaço tridimensional.

Tais heurísticas procuram subdividir uma topologia de Steiner em topologias de Steiner para subconjuntos dos pontos dados, de tal forma que cada topologia de Steiner seja cheia. Resolvem então o PSE para cada subconjunto com o algoritmo exato de Smith, após o que, reconcatenam as subárvores mínimas de Steiner de forma a encontrar uma árvore mínima de Steiner heurística para todo o conjunto de pontos dados.

A heurística Kalpakis-Ravada-Sherman[28][29] consiste na divisão do conjunto de pontos dados em subconjuntos nos quais são aplicados o método exato de Smith[9] por uma quatidade de tempo pré-determinada. Em seguida, as árvores encontradas para cada subconjunto são concatenadas de forma a obter uma árvore de Steiner, próxima da mínima, para o conjunto de pontos dados.

A divisão é feita em subconjuntos com no máximo p pontos e no mínimo três pontos do conjunto de pontos dados. Isto é, divide-se recursivamente o conjunto de pontos dados em dois subconjuntos separados pela mediana de uma de suas coordenadas com relação a todos os seus pontos, até que se tenham t conjuntos. À medida que t se aproxima de p, a solução melhora e o tempo de processamento da heurística aumenta.

Após a divisão do conjunto de pontos dados, o algoritmo de Smith é aplicado a cada um dos t subconjuntos, durante um tempo de CPU pré-determinado, retornando a menor árvore como uma solução heurística para o subconjunto.

Finalmente, os subconjuntos são reconcatenados através da construção de uma árvore geradora mínima para os pontos dados e os pontos de Steiner obtidos. Posteriormente é efetuado um pós-processamento da árvore com o intuito de retirar arestas excedentes.

Utilizando a mesma idéia da heurística anterior, a heurística de Smith-Weiss-Patel[30] utiliza o algoritmo de Smith para obter árvores de Steiner para subconjuntos de uma dada distribuição de pontos no $\Re^3$. Tal heurística difere da anterior pela forma como os subconjuntos são construídos: é utilizada a triangulação de Delaunay para construir tetraedros que, com o auxilio da árvore geradora mínima, são empilhados objetivando-se formar uma estrutura semelhante a uma *sausage*[1](o mesmo tipo de estrutura utilizada para conjecturar o ínfimo de ρ_3 em [1]). Em

seguida, o algoritmo de Smith é usado para minimizar as subestruturas obtidas pela combinação de tetraedros próximos.

Pereira[31] propôs três heurísticas que utilizam como base a mesma idéia central utilizada pela maioria das heurísticas aplicadas ao PSE em $\Re^3$ até o momento, ou seja: subdividir o conjunto de pontos dados em subconjuntos, aplicar o algoritmo de Smith em cada um desses subconjuntos e reconcatenar as estruturas obtidas a fim de encontrar uma árvore próxima da árvore mínima de Steiner.

Na primeira heurística, denominada Algoritmo Básico, a AGM construída para o conjunto de pontos dados é usada para encontrar subconjuntos, contendo de 3 até 4 pontos, que podem participar de uma subtopologia cheia. Em seguida, uma lista de prioridades é construída, armazenando as subárvores em ordem decrescente de redução do tamanho. Então, seguindo esta lista, de forma gulosa a heurística constrói a árvore final, através da concatenação destas subárvores, certificando-se sempre que uma concatenação não produza ciclos. A conexidade da árvore final é mantida pela inserção de arestas pertencentes a AGM. Esta heurística é executada em tempo linear, mas como necessita da construção de uma AGM, a sua complexidade é $O(p^2)$.

Pretendendo reduzir mais o tamanho da árvore obtida pelo Algoritmo Básico, Pereira apresenta um Algoritmo Melhorado, que difere do Algoritmo Básico na etapa final. Ao invés da conexidade da árvore ser obtida pela simples inserção de arestas pertencentes à AGM, este tenta obter subárvores com topologia cheia que conectem as subárvores desconectadas. Esta versão melhorada acarreta em uma redução significativa no tamanho da árvore final, sem que o tempo computacional aumente muito. A complexidade do Algoritmo Melhorado continua igual a do Algoritmo Básico, ou seja, $O(p^2)$.

Por fim, a terceira heurística desenvolvida por Pereira, seguindo a mesma estratégia dos Algoritmos Básico e Melhorado, utiliza como base para a divisão do conjunto de pontos dados em subconjuntos menores uma estrutura chamada de grafos de Gabriel, que corresponde a uma subárvore encontrada pela tetraedrização de Delaunay[32]. Utilizando a árvore final obtida através do Algoritmo Melhorado, a heurística inicia um procedimento iterativo com esta árvore. Neste procedimento, em cada iteração, a árvore é reconstruída com base na lista de prioridades, sendo

a árvore final encontrada menor que a árvore inicial. Devido ao uso do processo iterativo, a terceira heurística tem complexidade computacional igual a $O(p^3)$.

Para resolver problemas em dimensões maiores ou iguais a três, encontramos na literatura algumas propostas heurísticas e metaheurísticas.

Uma metaheurística simulated anneling[33] foi desenvolvida por Lundy [34], que implementou um algoritmo baseado na metaheurística *Simulated Annealing* [33] que utiliza dois tipos de perturbação (ou movimentos) durante o processo de *annealing*.

Uma das perturbações é realizada na topologia da árvore. Esta consiste em dada uma folha(ponto dado) e da árvore, remover as arestas b_{i1} e b_{i2} que, através do ponto de Steiner i, ligam os pontos 1 e 2 ao ponto e na árvore. Em seguida, na criação de uma aresta b_{12} para unir os pontos 1 e 2 e na remoção de uma aresta b_{34}. Por fim, as aresta b_{i3} e b_{i4} são criadas para conectar o ponto i ao ponto 3 e ao ponto 4. Este movimento é igual ao realizado por Thompson em [22] para obter árvores filogenéticas.

A outra perturbação é realizada na posição dos pontos de Steiner, onde são selecionados de um a $p-2$ pontos aleatoriamente, os quais terão a sua posição alterada para uma posição aleatória ou para uma posição tal que suas arestas adjacentes façam mutuamente um ângulo de 120^0.

Resultados computacionais são apresentados, no entanto não há informação sobre as instâncias utilizadas nos testes realizados em seu artigo.

Utilizando um modelo baseado em uma película de sabão, Montenegro [10] estendeu para dimensões maiores que dois a heurística de Relaxação Dinâmica desenvolvida por Chapeau-Blondeau et al[16]. Tal heurística, denominada Relaxação Dinâmica Estendida, simula a evolução de uma película de sabão sob tensões de forças superficiais.

A partir da árvore geradora mínima, construída para os pontos dados, uma árvore de Steiner é inicializada e, então, relaxada. Cada ponto de Steiner, um por vez, é movimentado na direção do gradiente relativo a soma do tamanho de suas arestas adjacentes.

Durante a relaxação, se uma aresta ligando dois pontos de Steiner tiver tamanho menor que um valor estipulado, é realizada uma troca entre as arestas adjacentes a estes pontos, escolhida com base no valor do gradiente referente a cada troca.

Esta heurística utiliza um parâmetro que controla o passo de cada ponto, em seu movimento na direção do gradiente, e um parâmetro que corresponde ao valor de referência para a troca de arestas pertencentes a pontos de Steiner muito próximos. Ambos os parâmetros decrescem no decorrer da heurística, que termina quando estes dois valores estiverem próximos de zero.

Esta heurística tem complexidade $O(p^2)$ e converge para uma solução mais rapidamente, quando comparada com os demais métodos desenvolvidos até o presente momento. Mesmo tendo bom desempenho, ou seja, encontrando soluções bem próximas das ótimas, esta heurística dificilmente encontra soluções ótimas.

Como a qualidade das soluções obtidas pela heurística de Relaxação Dinâmica Estendida depende de componentes aleatórias, um procedimento Multi-Start foi acrescentado, o qual reinicializa a heurística com uma semente da função geradora de números aleatórios diferente. A aplicação do procedimento Multi-Start aumenta o tempo computacional do algoritmo, mas melhora a qualidade das soluções encontradas. Esta heurística será mais detalhada em uma das seções seguintes, tendo em vista que propomos uma modificação que melhora o seu desempenho.

Partindo de uma árvore com topologia de Steiner cheia, Montenegro [10] desenvolveu uma busca local sobre as topologias de Steiner cheias possíveis para um conjunto de pontos dados. Nesta busca local, a vizinhança de uma topologia de Steiner cheia é definida como todas as topologias de Steiner cheias que podem ser obtidas pela troca de um par de arestas que conectam dois pontos de Steiner, também chamadas de arestas livres.

A metaheurística de Busca Tabu [35] foi implementada em conjunto com esta busca local para que fosse possível encontrar soluções de melhor qualidade.

Para melhorar a qualidade das soluções encontradas, foi adotado o mesmo procedimento de troca de arestas utilizado na heurística de Relaxação Dinâmica Estendida. Este mecanismo de diversificação foi acrescentado para compensar a vizinhança excessivamente restrita, já que a vizinhança tal como foi definida não mapeia o espaço de todas as topologias de Steiner cheias possíveis para um conjunto de pontos dados. Resultados computacionais mostraram que para um conjunto de poucos pontos a Busca Tabu, em um tempo de CPU bem elevado, retorna soluções comparáveis com as encontradas pela heurística de Relaxação Dinâmica Estendida.

Com relação aos Algoritmos Evolutivos, um Algoritmo Genético básico foi desenvolvido por Montenegro e Maculan[36] considerando os vetores topologias como os cromossomos, já que um vetor topologia possui uma relação biunívoca com as topologias de Steiner cheias para um conjunto de pontos dados. A razão de Steiner após a minimização da árvore relacionada ao vetor topologia foi utilizada como função objetivo.

A população inicial é composta por um conjunto de soluções obtidas com a inserção de pontos de Steiner na árvore geradora mínima construída para os pontos dados, tal como na etapa de inicialização da heurística de Relaxação Dinâmica Estendida. Os operadores de cruzamento e mutação deste algoritmo são aplicados da mesma forma que em um Algoritmo Genético clássico, observando que cada operação deve resultar em um vetor topologia. Ou seja, as operações de reprodução e mutação não destroem a correspondência de um cromossomo a um vetor topologia.

Após um número máximo de iterações sem melhora da solução, o algoritmo termina. Os resultados computacionais apresentados no trabalho mostram que esta heurística tem um grande potencial, tendo em vista que mesmo em sua versão básica conseguiu para uma grande quantidade de instâncias encontrar soluções iguais as retornadas pelo algoritmo exato.

Montenegro et al. [10] desenvolveram uma heurística de busca local baseada em vetores topologia. Esta foi possivelmente a primeira proposta de utilização direta desses vetores em uma heurística de busca local para o problema em dimensão $n > 2$. Entretanto, é importante destacar que uma busca local diferente, mas também baseada em vetores topologia, já havia sido desenvolvida anteriormente por Hürlimann [37] para o problema no plano, tendo sido reportados resultados de boa qualidade.

Na heurística de busca local proposta por Montenegro et al.[10], a vizinhança é definida a partir de todos os vetores topologia que podem ser obtidos pela troca de uma de suas componentes. Mais à frente, nesta tese, esta busca local baseada em vetores topologia será mais detalhada. A seguir, daremos alguns detalhes de dois outros algoritmos que incorporam esta busca local em vetores topologia: um algoritmo de Otimização Microcanônica e um algoritmo GRASP.

O algoritmo de Otimização Microcanônico foi aplicado por Montenegro et al.

em [38] para estender a busca local em vetores topologia além do primeiro mínimo local encontrado. Partindo de uma solução inicial igual a usada na heurística de Relaxação Dinâmica Estendida, o algoritmo alterna entre um procedimento chamado de iniciação e o outro de *sampling*.

O procedimento de iniciação consiste na simples aplicação da busca local, resultando em um mínimo local. Objetivando-se continuar a busca por soluções melhores que o mínimo local encontrado, o procedimento de *sampling* é iniciado com o mínimo local, sobre o qual são realizadas pequenas modificações que serão aceitas dentro de um intervalo de energia associado à simulação microcanônica de Creutz[39]. Estes dois procedimentos são executados iterativamente, utilizando sempre a solução obtida em uma iteração anterior como a solução inicial da próxima, até que um critério de parada seja satisfeito.

Comparada com as heurísticas apresentadas até agora, segundo resultados computacionais, esta foi a que produziu os resultados mais próximos do ótimo quando não o próprio ótimo.

Recentemente, no trabalho de Rocha[18], um Algoritmo GRASP[40] combinado com Path-Relinking [41] foi aplicado ao Problema de Steiner Euclidiano em $\Re^n$ utilizando a busca local em vetores topologia. Resultados computacionais foram obtidos com implementações deste algoritmo em ambiente sequencial e paralelo.

Com relação a qualidade das soluções encontradas pelo experimento computacional, temos que este produz árvores de Steiner de menor tamanho quando comparadas as encontradas pelos demais algoritmos heurísticos citados nesta revisão, independente de qual ambiente o algoritmo foi implementado.

Já quando comparados os tempos de CPU, nota-se uma diferença entre os algoritmos implementados nos diferentes ambientes, sendo o algoritmo implementado em ambiente híbrido o que produz soluções com menor tempo de CPU.

2.7 O Método de Smith

A solução exata do Problema de Steiner Euclidiano em $\Re^n$ pode ser encontrada por um algoritmo proposto por W. D. Smith em [9], que enumera todas as árvores de Steiner possíveis para o conjunto de pontos dados, utilizando uma estrutura

chamada vetor topologia. Tal estrutura corresponde a um vetor de $p-3$ componentes para representar cada topologia de Steiner cheia para os p pontos dados de forma biunívoca. Ou seja, dada uma topologia de Steiner cheia, podemos representá-la de maneira única com um vetor topologia, como também podemos obter uma única topologia de Steiner através de um vetor topologia. A cada nó da enumeração, uma topologia de Steiner cheia é minimizada pela resolução de n sistemas de equações lineares iterativamente, com o objetivo de encontrar a árvore mínima relativa associada a ela.

Para minimizar o número de topologias a serem consideradas no processo de enumeração, são realizados cortes baseados em estruturas parciais do vetor topologia, eliminando grupos de topologias de Steiner cheias que resultam em árvores mínimas relativas cujo tamanho é maior que a atual melhor árvore.

2.7.1 Vetores Topologia

A estrutura denominada vetor topologia consiste em um vetor de $p-3$ componentes, onde é selecionado para cada posição i um valor, a_i, tal que $1 \leq a_i \leq 2i+1$.

Obtemos um vetor topologia construtivamente a partir de um vetor nulo, o qual representa uma topologia trivial envolvendo três pontos selecionados dentre os p pontos dados, que denotamos respectivamente por x_1, x_2 e x_3, e um ponto de Steiner $p+1$. As arestas desta topologia trivial são numeradas de 1 a 3, de acordo com os pontos aos quais são adjacentes.

Em seguida, incluímos um quarto ponto, x_4, fazendo a conexão deste com o ponto de Steiner $p+2$, que é inserido em uma das três ligações da topologia anterior. Esta inclusão dará origem a duas novas ligações, do quarto ponto ao ponto $p+2$ (ligação 4) e deste ao ponto $p+1$ (ligação 5). Este processo é então repetido para incluir, seqüencialmente, todos os demais $p-4$ pontos dados. Para a inserção de um ponto x_i qualquer, onde $i \geq 4$, serão criadas duas novas arestas, ambas ligadas ao ponto de Steiner $p+i-2$, sendo a aresta $2i-4$ ligada ao ponto x_i e a aresta $2i-3$ ligada a um ponto de Steiner.

A cada inserção de um ponto dado à topologia de Steiner, um número associado a cada ligação escolhida para inserção do ponto de Steiner é armazenado no vetor topologia, na posição relativa à quantidade de pontos inseridos.

Na figura 2.13 podemos observar as etapas de construção do vetor topologia para uma topologia de Steiner cheia com cinco pontos dados.

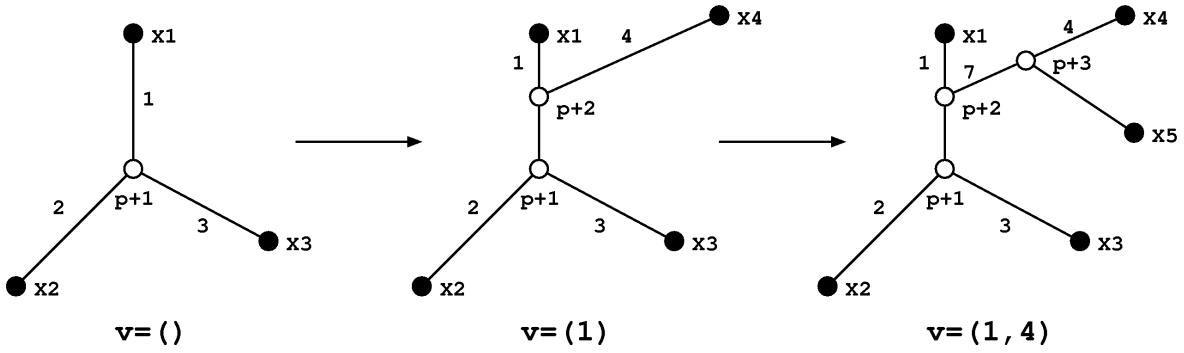

Figura 2.13: Construção de um vetor topologia com cinco pontos dados

2.7.2 Minimização das Topologias

Para minimizar a função não-diferenciável associada ao tamanho da árvore com relação à posição dos pontos de Steiner em uma dada topologia, Smith propôs um método iterativo similar ao método de Gauss-Seidel. Tal método produz uma sequência de pontos de Steiner cujas coordenadas tendem à posição ótima.

Sendo T um vetor topologia, uma iteração deste método é definida pela resolução do seguinte sistema de equações lineares:

$$\vec{x}_k^{(i+1)} = \frac{\sum_{j|(jk)\in T} \frac{\vec{x}_j^{(i+1)}}{\|\vec{x}_j^i - \vec{x}_k^i\|}}{\sum_{j|(jk)\in T} \frac{1}{\|\vec{x}_k^i - \vec{x}_j^i\|}},\ p+1 \leq k \leq 2p-2. \tag{2.18}$$

Onde o vetor $\vec{x}_k$ é um vetor formado pelas coordenadas do k-ésimo ponto dado ou ponto de Steiner.

Cada uma destas iterações corresponde, de certa forma, a relaxação de um sistema de molas ideais. Neste sistema, substituímos cada ligação $[i, j]$ da topologia T, referente à árvore a ser minimizada, por uma mola ideal cuja força de restauração $F_{(i,j)}$ ao deslocarmos um ponto de Steiner x_j com relação ao ponto x_i, em uma iteração k, é proporcional ao inverso da distância entre estes pontos , ou seja,

$$F_{(i,j)}^k = \frac{2(\vec{x}_i - \vec{x}_j)}{\|\vec{x}_i^k - \vec{x}_j^k\|}$$

A energia potencial de cada mola deste sistema é igual à

$$E_{(i,j)}^k = \frac{\|\vec{x}_i - \vec{x}_j\|^2}{\|\vec{x}_i^k - \vec{x}_j^k\|},$$

dessa forma, para $S = \{\vec{x}_{p+1}, \vec{x}_{p+2}, ..., \vec{x}_{2p-2}\}$ o conjunto de pontos de Steiner, a energia potencial do sistema de molas ideais é

$$E^k(S) = \sum_{(i,j)\in T, j\geq p+1, i<j} \frac{\|\vec{x}_i - \vec{x}_j\|^2}{\|\vec{x}_i^k - \vec{x}_j^k\|}.$$

Em uma iteração k deste método iterativo, temos que, para $S^k = \{\vec{x}_{p+1}^k, \vec{x}_{p+2}^k, ..., \vec{x}_{2p-2}^k\}$, a energia potencial do sistema é igual a

$$E^k(S^k) = \sum_{(i,j)\in T, j\geq p+1, i<j} \frac{\|\vec{x}_i^k - \vec{x}_j^k\|^2}{\|\vec{x}_i^k - \vec{x}_j^k\|} = \sum_{(i,j)\in T, j\geq p+1, i<j} \|\vec{x}_i^k - \vec{x}_j^k\|,$$

Como o tamanho da árvore, para o conjunto S, é igual a

$$L(S) = \sum_{(i,j)\in T, j\geq p+1, i<j} \|\vec{x}_i - \vec{x}_j\|,$$

então podemos afirmar que

$$E^k(S^k) = L(S^k) = \sum_{(i,j)\in T, j\geq p+1, i<j} \|\vec{x}_i^k - \vec{x}_j^k\|,$$

e que

$$\nabla L(S^k) \propto \nabla E^k(S^k)$$

No trabalho de Smith [9] foi provado um teorema que garante a convergência da sequência das posições dos pontos de Steiner , obtidas através deste método iterativo, para a posição tal que o tamanho da árvore é mínimo para a sua topologia.

Com o objetivo de contornar a não-definição do sistema de equações lineares nos casos em que árvores possuem ligações cujo tamanho é nulo, um valor de erro bem próximo de zero substitui o valor do tamanho destas ligações. Dessa forma, a árvore encontrada ao final do método iterativo será tão próxima da árvore mínima relativa quanto o valor de erro estiver próximos de zero, caso a topologia seja degenerada.

Em cada iteração, os n sistemas de equações lineares $(p-2)$x$(p-2)$ são solucionados com $O(pn)$ *flops* pelo método de eliminação de Gauss, já que as matrizes associadas aos sistemas tendem a ser tridiagonais.

Para detectar a convergência do método iterativo, utiliza-se a medida de erro definida abaixo

$$E^2 = \sum_{(i\in S, ij\in T, ik\in T, j\neq k)} Max(2(\vec{x}_j - \vec{x}_j) + \|\vec{x}_j - \vec{x}_i\|.\|\vec{x}_k - \vec{x}_i\|, 0), \tag{2.19}$$

na qual, à medida que a árvore se aproxima da árvore mínima relativa para a topologia T, o ângulos entre qualquer par de arestas é maior ou igual a 120^o, e consequentemente o valor de E torna-se bem menor que o tamanho da árvore $L(S)$.

2.7.3 Enumeração das Topologias

Este é o procedimento principal do algoritmo, no qual um vetor topologia é construído componente a componente. Para cada elemento inserido, o vetor topologia resultante é minimizado, até que o mesmo tenha p-3 componentes. A minimização de estruturas parciais é fundamental para a enumeração, pois se uma estrutura parcial tem o tamanho da árvore, maior que a menor árvore encontrada até o momento, então qualquer estrutura obtida com esta estrutura parcial como prefixo também possuirá valor maior. Logo, a enumeração não é continuada com esta estrutura parcial, e um procedimento backtracking é realizado para eliminar famílias de estruturas ruins associadas a ela.

Capítulo 3

Algoritmos Propostos

Em função da alta complexidade para encontrar soluções exatas para o Problema de Steiner Euclidiano em $\Re^n$, faz-se necessário a construção de algoritmos baseados em heurísticas e metaheurísticas com a expectativa de produzir soluções de boa qualidade em um tempo computacional razoável.

Neste capítulo, apresentaremos a heurística de Relaxação Dinâmica Estendida utilizada para resolver o PSE para dimensões $n \geq 3$. Em seguida, apresentaremos uma modificação realizada nesta heurística com a finalidade de melhorar a qualidade das soluções produzidas. Também apresentaremos algoritmos que combinam a metaheurística ILS com a busca local em vetores topologia.

3.1 Relaxação Dinâmica Estendida

A heurística de Relaxação Dinâmica, desenvolvida por Chapeau-Blondeau et al.[16], teve como objetivo obter soluções heurísticas do Problema de Steiner Euclidiano para pontos pertencentes ao plano. Posteriormente, adaptações foram efetuadas por Montenegro et al., em [10], com o intuito de aplicar esta heurística, sob o nome de Relaxação Dinâmica Estendida, também em distribuições de pontos em espaços euclidianos de dimensão maior que dois.

Essa heurística, baseada no modelo da película de sabão[14][15], é apresentada na seção de formulações do segundo capítulo. Simula a dinâmica dessa película sujeita tensões de forças superficiais, procurando uma configuração estável que aproxima uma árvore mínima de Steiner para os pontos dados.

Para tanto, cada ponto de Steiner é tratado de maneira semelhante a um autômato celular, sendo a sua posição modificada através de informações obtidas somente pelos pontos que compartilham uma ligação com ele em uma determinada topologia. A informação usada para obter a posição ótima do ponto de Steiner é dada, para uma topologia de Steiner T, como o gradiente da função

$$F(x_i) = \sum_{j|(ij)\in T} \|x_i - x_j\| \ , \tag{3.1}$$

a qual é a soma do tamanho de cada uma das arestas adjacentes ao ponto de Steiner x_i. Se x_a, x_b e x_c são os pontos adjacentes ao ponto de Steiner x_i, o gradiente de $F(x_i)$ é portanto igual a

$$\nabla F(x_i) = -\frac{(x_a - x_i)}{\|x_a - x_i\|} - \frac{(x_b - x_i)}{\|x_b - x_i\|} - \frac{(x_c - x_i)}{\|x_c - x_i\|}. \tag{3.2}$$

Quando o ponto de Steiner estiver localmente em uma posição na qual o tamanho de suas arestas é mínimo, ou seja, quando $\nabla F(x_i) = 0$, o ponto estará em uma posição estável. E consequentemente, suas arestas adjacentes terão ângulos mútuos de 120^0.

Com isso, a cada iteração um ponto de Steiner é reposicionado conforme a equação:

$$x_i^{(k+1)} = x_i^k - \delta \nabla F(x_i^k). \tag{3.3}$$

O parâmetro δ é chamado de coeficiente de proporcionalidade, sendo utilizado para atenuar a oscilação da posição do ponto de Steiner quando este está proximo de sua posição ótima.

Um procedimento denominado evolução atualiza iterativamente cada um dos pontos de Steiner, em ordem aleatória, com base na equação acima. O valor de δ é progressivamente reduzido e, ao atingir um valor próximo de zero, determina o critério de parada da heurística. Na figura 3.1 podemos observar a atualização de um ponto de Steiner durante o procedimento de evolução.

O procedimento de evolução não é suficiente para encontrar uma árvore mínima de Steiner, pois esse só encontra uma árvore mínima relativa para uma topologia previamente fixada, que pode não ser a topologia de uma solução ótima. Dessa forma, é necessário que a topologia possa ser modificada, tal como ocorre com uma película de sabão no decorrer do experimento físico. Para tanto, é criado um procedimento

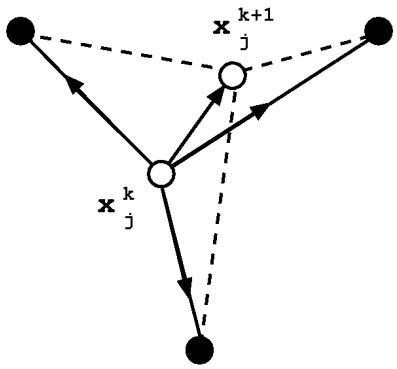

Figura 3.1: Procedimento de evolução

denominado interação. Esse procedimento permite que, sempre que o tamanho de uma ligação entre dois pontos de Steiner for menor ou igual a um parâmetro η uma troca entre suas arestas adjacentes possa ser realizada. As trocas possíveis entre as arestas adjacentes são avaliadas medindo o valor do gradiente dos pontos de Steiner para cada troca. Sendo efetivada a troca que resultará em gradientes de maior magnitude.

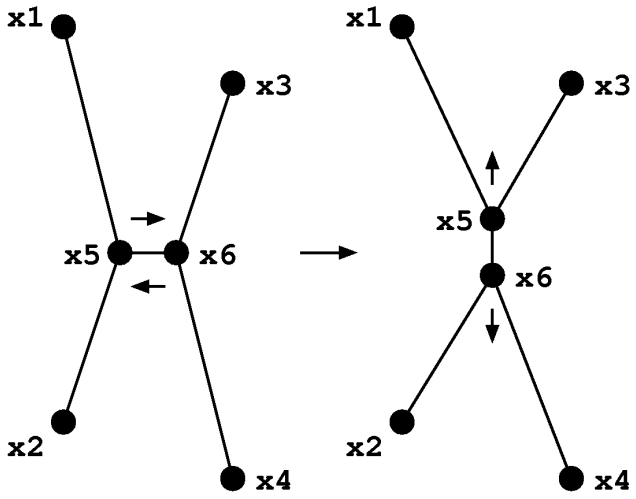

Figura 3.2: Procedimento de Interação aplicado a dois pontos de Steiner.

Na figura 3.2, o gradiente é calculado para as duas topologias obtidas ao efetuarmos as trocas de arestas entre os pontos de Steiner x_5 e x_6, que resultam em topologias diferentes da original. Neste caso, as trocas resultariam nas seguintes topologias $\{(x_1, x_5), (x_3, x_5), (x_2, x_6), (x_4, x_6)\}$ e $\{(x_1, x_6), (x_3, x_5), (x_2, x_5), (x_4, x_6)\}$. A topologia escolhida é aquela cujo o vetor gradiente tem maior magnitude.

O desempenho do algoritmo depende da árvore inicial à qual serão aplicados os

procedimentos de evolução e de interação. Esta árvore é construída através de um procedimento de inicialização, o qual utiliza como base a árvore geradora mínima para os pontos dados.

Para cada ponto i com mais de uma aresta adjacente na árvore geradora mínima construída para os pontos dados, o procedimento de inserção de pontos de Steiner seleciona um par de arestas, (i, a) e (i, b), adjacentes a i. Sendo então inserido um ponto de Steiner $p+1$ na árvore, através da criação da aresta $(i, p+1)$, e substituindo as arestas (i, a) e (i, b) pelas arestas $(p + 1, a)$ e $(p + 1, b)$.

Este processo prossegue com a inserção de outros pontos de Steiner na árvore geradora mínima, para cada par de arestas restante adjacentes a i, até que este tenha grau 1. O processo é aplicado a todos os pontos da árvore com grau maior do que um, com exceção dos pontos de Steiner inseridos. A figura 3.3 ilustra a construção de uma árvore inicial para seis pontos dados.

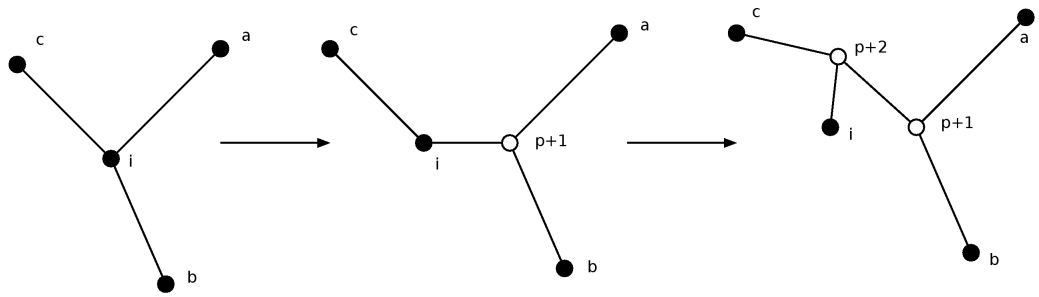

Figura 3.3: Inserção de pontos de Steiner em uma Árvore Geradora Mínima

Podemos observar, pelo processo de construção apresentado na figura 3.3, que diferentes escolhas da ordem em que as arestas são selecionadas para inserção dos pontos de Steiner levam a diferentes topologias de Steiner. Ou seja, não existe unicidade para a árvore encontrada por esse processo de construção. No entanto, o conjunto de todas as topologias de Steiner cheias diferentes que podem ser encontradas por esta construção é finito, uma vez que cada topologia esta relacionada a uma permutação da ordem em que são escolhidos pares de arestas para serem inseridos os pontos de Steiner.

O algoritmo completo executa os procedimentos de evolução e interação repetidas vezes, partindo da árvore inicial obtida pela inserção de pontos na AGM e relaxando-a até que a árvore resultante esteja próxima de uma árvore relativa mínima para os pontos dados. Como a cada iteração completa, cada ponto de Steiner é alterado

uma única vez, temos que esse processo é executado em $O(np)$. Entretanto, como a relaxação é realizada sobre uma árvore construída a partir da AGM, temos que a complexidade total da heurística é $O(np^2)$, já que em dimensões maiores que dois o melhor algoritmo encontra a AGM em $O(np^2)$.

A construção da árvore inicial, tal como idealizada por Chapeau-Blondeau et al.[16], percorre a AGM p vezes à procura dos vizinhos de cada um dos pontos dados, implicando em uma complexidade de $O(p^2)$. No entanto, dada uma ordenação dos pontos da AGM, a complexidade desta etapa é reduzida ao se construir uma matriz M tal que cada elemento m_{ij} armazena o j-ésimo filho do i-ésimo ponto dado. Com isso esta matriz possuirá, no plano, p linhas e 6 colunas, já que o número de filhos de um ponto pertencente a uma AGM construída para pontos pertencentes ao $\Re^2$ é de, no máximo, seis.

Porém, à medida que a dimensão dos pontos aumenta, o número máximo possível de filhos de um ponto da AGM cresce muito. Observando o *kissing number* τ_n em dimensão n (corresponde ao número máximo possível de hiperesferas que tocam a superfície de uma hiperesfera, em dimensão n, quando todas tem o mesmo tamanho), é fácil verificar[10] que o grau máximo, g_n, de um ponto pertencente ao $\Re^n$ em uma AGM é tal que:

$$min(\tau_n, p-1) \leq g_n \leq p-1 \ .$$

No entanto, limitantes assintóticos mostram que o crescimento de τ_n em relação a n é exponencial[4], tornando inviável a construção da matriz M para o PSE em $\Re^n$ quando $n >> 2$.

Para tornar viável a aplicação dessa heurística ao PSE em dimensões maiores que dois, uma estrutura de dados composta por três vetores foi criada de forma a armazenar as informações, da árvore geradora mínima. Tais informações são necessárias para a construção da árvore inicial em tempo e espaço em memória lineares.

Uma variável σ_p, que representa uma unidade natural de tamanho, ou seja, uma medida de separação característica dos pontos dados, é utilizada para calibrar os parâmetros iniciais δ e η como proporcionais a σ_p, que é dado como

$$\sigma_n = p^{-\frac{1}{n}} \tag{3.4}$$

Devido à não unicidade da árvore inicial e à aleatoriedade na ordem em que os

pontos de Steiner são movimentados, uma metaheurística Multi-Start[42] foi aplicada à heurística de Relaxação Dinâmica Estendida. Tal heurística é reiniciada com diferentes sementes do gerador de números pseudo-aleatórios, de forma a executar a heurística com árvores iniciais diferentes e com diferentes ordens de escolha dos pontos a serem relaxados. Possibilitando, assim, encontrar árvores com tamanho menor que a encontrada por uma única execução desta heurística.

Resultados computacionais obtidos na literatura[42] mostram que a Relaxação Dinâmica Estendida é a heurística mais rápida e robusta dentre todas as propostas para o Problema de Steiner Euclidiano em $\Re^n$, podendo ser aplicada a distribuições com mais de 100.000 pontos dados em tempo computacional relativamente baixo. Porém, esta heurística dificilmente obtém soluções ótimas, mesmo distribuições com poucos pontos dados. A sua versão Multi-Start é capaz de encontrar algumas poucas soluções ótima às expensas de um tempo computacional bem significativo.

3.2 Melhoria Realizada na Relaxação Dinâmica Estendida

Nesta trabalho, implementou-se uma pequena modificação na heurística de Relaxação Dinâmica Estendida objetivando-se encontrar soluções melhores que as já obtidas pela heurística e sua versão Multi-Start,mas mantendo sua robustez e sua reduzida exigência de tempo computacional.

A modificação realizada na heurística está relacionada com a inserção de informação global no procedimento de relaxação. Conforme já observado, na etapa de evolução, os pontos de Steiner são movimentados para posições localmente ótimas sem garantia de que, ao final do processo de relaxação, a árvore encontrada será uma árvore de Steiner.

Para garantir que a árvore construída pela heurística seja uma árvore de Steiner, utilizamos nesta heurística o procedimento de minimização de árvores contido no algoritmo exato de Smith.

No entanto, essa visão local do procedimento de evolução permite que o procedimento de interação modifique a topologia da árvore a cada reposicionamento dos pontos de Steiner. Dessa forma, o procedimento de minimização de árvores

de Smith[9] é aplicado a topologia final retornada pela heurística de Relaxação Dinâmica Estendida.

A simples execução deste procedimento fez com que a solução final encontrada melhorasse drasticamente, de tal modo que, para 1000 distribuições contendo 10 pontos dados cada pertencentes ao $\Re^3$, o algoritmo estendido padrão não obteve soluções ótimas, enquanto que o algoritmo melhorado encontrou 461 soluções ótimas.

Os resultados são ainda melhores quando aplicamos a metaheurística Multi-Start ao algoritmo melhorado. Para o mesmo conjunto de instâncias acima, o algoritmo Multi-Start com procedimento padrão obteve 2 soluções ótimas em 0.05142 segundo, enquanto que a versão Multi-Start da heurística melhorada obteve 788 soluções ótimas em 0.07019 segundo.

3.3 Busca Local Iterativa Aplicada ao PSE

Nesta seção iremos descrever os algoritmos propostos neste trabalho para solucionar o Problema de Steiner Euclidiano em $\Re^n$ utilizando a metaheurística Busca Local Iterativa [43]*(ILS - Iterated Local Search)*.

Antes de apresentarmos detalhes sobre os algoritmos propostos, iremos descrever em termos gerais o principio básico da metaheurística ILS e seus componentes principais.

3.3.1 Busca Local Iterativa

A Busca Local Iterativa é uma metaheurística que tem sido aplicada com sucesso a diversos problemas da literatura tais como o Problema do Caxeiro Viajante[44], Particionamento de Grafos[45], Problema das k-medianas[46], entre outros. Esta metaheurística pode ser encontrada sob outras nomenclaturas como *iterated descent*[47], *large-step Markov chain*[44], *iterated Lin-Kernighan*[48] e *chained local optimization*[45]. Porém, atribuimos a sua origem ao trabalho de Baxter[49] publicado em 1981.

Do ponto de vista teórico, por exemplo, consideremos um problema de otimização C, onde S é o seu espaço de soluções viáveis, podendo este ser discreto quando C é um problema de otimização combinatória, ou contínuo quando o problema é de

otimização contínua. Definimos $f : S \rightarrow \Re^+$ como o custo de cada elemento $s \in S$, onde o objetivo do problema é encontrar uma solução $s^* \in S$ tal que $f(s^*) \leq f(s)$ $\forall s \in S$, ou seja, o mínimo da função f.

Normalmente, as heurísticas construídas especificamente para C tendem a encontrar uma solução viável que é um mínimo local para o problema. Para ilustrar melhor o mecanismo básico da metaheurística ILS adotaremos a heurística de Busca Local aplicada a C (Obs.: No entanto, pode ser aplicada qualquer heurística específica para o problema).

Começando de uma solução inicial $s^0 \in S$, a heurística de busca local procura por uma solução que reduza o valor de $f(s^0)$ em um subconjunto $N(s^0) \subseteq S$ formado por todas as soluções que podem ser obtidas de s^0 pela modificação de um de seus atributos. Este subconjunto é chamado de vizinhança de uma solução. Após encontrar uma solução $s^1 \in N(s^0)$ tal que $f(s^1) \leq f(s^0)$ a busca continua na vizinhança de s^1, e assim por diante, até que em uma iteração j seja encontrado $f(s^j) < f(s)$ $\forall s \in N(s^j)$, ou seja, s^j é um mínimo local para o problema C.

Em geral, as heurísticas desenvolvidas para um problema C encontram um mínimo local em pouco tempo computacional. Para encontrar soluções melhores do que as obtidas pelas heurísticas, observando a dependência da solução inicial, métodos que reiniciam a busca local com diferentes soluções iniciais, denominados Multi-Start[50], podem ser utilizados (tal como na proposta da subseção anterior). Porém, a probabilidade de encontrar soluções melhores usando esses métodos diminui à medida que o tamanho da instância do problema C aumenta[43].

Seja $s \in S$ uma solução a partir da qual, após a aplicação de uma busca local, é obtido o mínimo local s^*. Definimos o conjunto S^* tal que $\forall s \in S$ temos $s^* \in S^*$. Já que os valores pertencentes a S^* correspondem aos mínimos locais obtidos a partir dos elementos $s \in S$ espera-se, que em média, os valores $s^* \in S^*$ apresentem $f(s^*) \leq f(s)$. Com isso, se pudermos efetuar uma busca local sobre o conjunto S^*, esperamos encontrar soluções menores que as obtidas pela aplicação da busca local em S . Aplicando de forma recursiva a busca local em cada conjunto de mínimos locais obtidos, esperaremos encontrar soluções cada vez mais próximas da solução exata do problema.

No entanto, necessitaremos de uma nova estrutura de vizinhança para cada sub-

conjunto de mínimos locais sobre o qual será aplicada a busca local. De um ponto de vista canônico podemos definir uma vizinhança em S^* como: Se s^{*1} e s^{*2} pertencentes a S^* são vizinhos, então estes possuem soluções vizinhas próximas em S. Para encontrar soluções vizinhas em S^*, uma sequência aleatória de vizinhos próximos é gerada. Sobre esssa sequência são realizadas buscas locais, a partir de cada um de seus elemetos, até que, para um s^j, o seu mínimo local s^{*j} seja diferente de todos os mínimos locais obtidos pela aplicação da busca local a partir de outros elementos da sequência, $s^{*j} \neq s^*$.

E fácil entender porque o processo de efetuar buscas locais recursivamente em subconjuntos de mínimos locais de S implica no consumo de um elevado tempo computacional. Isso se deve ao fato de que são necessárias inúmeras buscas locais só para determinar um vizinho no conjunto de mínimos locais S^*.

Para contornar essa dificuldade, uma noção mais fraca de vizinhança é utilizada para realizar a busca local em S^*. Partindo de uma solução $s \in S$ com mínimo local $s^* \in S^*$ obtido pela busca local, uma perturbação em s^* é feita esperando que $s' \in S$ esteja em uma região de atração de um outro mínimo local, ou seja, ao ser aplicada a busca local em s' o mínimo local encontrado $s^{*'}$ é diferente de s^*. Se $s^{*'}$ for aceito por um procedimento de aceitação, este torna-se a solução corrente. O processo de perturbação, busca local e aceitação são aplicados iterativamente até que um critério de parada seja atingido.

Em algumas implementações da metaheurística ILS são utilizadas informações adiquiridas ao longo das iterações para auxiliar os procedimentos de perturbação e aceitação.

A figura 3.4 contém uma ilustração que representa o processo de busca local iterativa e na figura 1 é apresentado o pseudocódigo simplificado desta metaheurística.

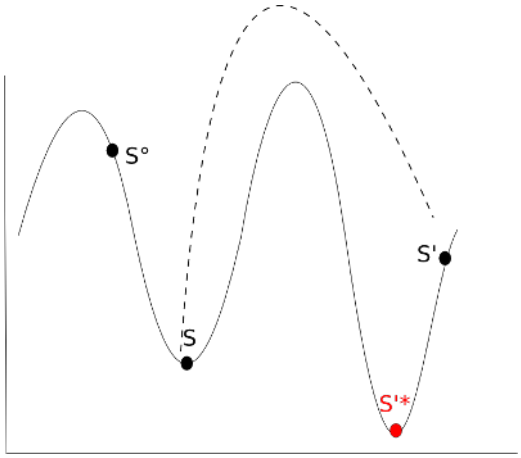

Figura 3.4: Iteração básica do ILS. Uma solução s^* é perturbada e a busca local é aplicada a sua perturbação s', encontrando uma solução menor s'^*.

$s^0 \leftarrow$ Inicialização();
$s \leftarrow$ Busca-local(s^0);
repita
  $s' \leftarrow$Perturbação(s);
  $s'^* \leftarrow$Busca-local(s');
  $s \leftarrow$Aceitação(s, s'^*);
até *Critério de parada*;

Algoritmo 1: Busca Local Iterativa - ILS

3.3.2 Aplicação ao Problema de Steiner Euclidiano em $\Re^n$

Nesta seção, detalharemos como os procedimento de inicialização, busca local, perturbação e aceitação, contidos no algoritmos ILS, que foram implementados para a resolução do Problema de Steiner Euclidiano em R^n.

Inicialização

O procedimento de inicialização é responsável pela construção da solução viável inicial. Tal solução é usada como ponto de partida para que os demais procedimentos possam construir uma trajetória desta solução até a solução viável de melhor

qualidade encontrada pela heurística.

A solução inicial pode ser obtida de forma aleatória, ou seja, pode ser qualquer solução viável do conjunto de solução possíveis. No entanto, segundo [43], o algoritmo ILS necessitará de um tempo consideravelmente menor para encontrar uma solução de boa qualidade caso utilize uma solução inicial construída com base em um heurística de construção gulosa.

Para o Problema de Steiner Euclidiano em $\Re^n$, dois procedimentos de obtenção da solução inicial viável foram implementados. O primeiro é o procedimento de construção guloso da fase de construção do algoritmo GRASP, desenvolvido para o PSE em [18]. Nesse procedimento, o parâmetro alpha associado à lista de candidatos restrita (LCR), foi definida como zero. O segundo procedimento é igual ao procedimento de inserção de pontos de Steiner na AGM utilizado pela heurística de Relaxação Dinâmica Estendida para obter uma topologia de Steiner inicial.

A partir de um vetor topologia nulo, o procedimento de construção guloso minimiza as topologias de Steiner relacionadas a todos os vetores topologia de tamanho um, selecionando aquele associado à árvore de Steiner com menor tamanho. Em seguida, um vetor topologia com duas componentes é construído utilizando como prefixo o vetor topologia de tamanho um obtido anteriormente. O segundo elemento do vetor é escolhido de tal forma que a árvore relativa mínima associada tenha o menor tamanho. Continuamos assim a construção, sempre selecionando o vetor topologia com i componentes que resulte na árvore relativa mínima de menor tamanho dentre aquelas cujos vetores topologia tenham como prefixo as mesmas $i-1$ componentes do vetor topologia encontrado na iteração anterior. Este processo continua até que um vetor com $p-3$ componentes seja encontrado.

A solução inicial obtida pela inserção de pontos de Steiner na AGM tem a vantagem de produzir sempre topologias de Steiner que, ao serem minimizadas, levam a árvores relativas mínimas com tamanho menor ou igual ao da árvore geradora mínima. Entretanto, por apresentar uma solução inicial de ótima qualidade, esta pode dificultar a busca de soluções melhores pelo procedimento de busca local em sua vizinhança, bem como fazer com que a perturbação não tenha efeito em transferir a busca para uma região com soluções melhores. Já a solução inicial obtida pela construção gulosa nem sempre resulta em uma árvore de Steiner menor que a

árvore geradora mínima, sendo uma solução de qualidade ruim. Contudo, por outro lado, esta pode ser um excelente ponto de partida para que os procedimentos da metaheurística ILS encontrem soluções de boa qualidade.

Busca Local em Vetores Topologia

Desenvolvemos a metaheurística ILS com base na heurística de busca local proposta por Montenegro et al.[10] para o PSE, a qual utiliza a estrutura de vetores topologia criada por Smith em seu método exato.

Nessa busca local, consideramos um vetor topologia v como uma solução viável para o problema, já que podemos relacionar de forma única cada vetor topologia com uma árvore mínima relativa obtida pela minimização da topologia de Steiner cheia associada a ela. Desta forma, o tamanho de uma árvore mínima relativa será o custo do vetor topologia associado a ela ($f(v)$). Com isso, o conjunto de soluções viáveis do problema torna-se igual ao espaço definido por todos os vetores topologia.

Desta forma, definimos a estrutura de vizinhança $N(v) \subseteq V$ de um solução $v \in V$ como o conjunto de todos os vetores topologia que podem ser obtidos pela alteração do valor de uma de suas componentes $v(i)$,onde $1 \leq i \leq (p-3)$, para um valor $1 \leq j \leq 2i-1$.

Dado um vetor topologia v, a busca local procura por um vetor topologia v' pertencente à vizinhança $N(v)$, tal que $f(v') \leq f(v)$. Para isso, escolhe-se aleatoriamente uma componente i dentre as $p-3$ componentes do vetor topologia v e um valor j inteiro, diferente de $v(i)$, pertencente ao intervalo $[1, 2i+1]$. Sendo o vetor topologia v' tal que $v' = v$ e $v'(i) = j$. E se $f(v') \leq f(v)$, então efetuamos o movimento $v \leftarrow v'$ e $f(v) \leftarrow f(v')$. Com isso, a busca local prossegue, com a solução corrente v, procurando por um vetor topologia na vizinhança $N(v)$ que leve a uma árvore mínima relativa com tamanho menor. Caso $f(v') > f(v)$, um novo vetor topologia $v' \in N(v)$ é criado.

Após a geração de uma certa quantidade de vetores topologia com custo maior que o melhor custo obtido pela busca até o momento, na vizinhança de v, a busca local é encerrada. Nesse caso, v é definido então como um mínimo local.

Como a quantidade de valores possíveis em cada componente i de um vetor topologia é $2i+1$, temos que o número de vizinhos de um vetor topologia que

podem ser obtidos na busca local é igual a $\sum_{i=1}^{p-3} 2i + 1$, ou seja, $O(p^2)$.

Nas heurísticas de busca local em vetores topologias implementadas na literatura, o número máximo de vizinhos considerados na busca varia no intervalo $[2p, 5p]$. Como o número de vizinhos de um vetor topologia é $O(p^2)$, temos que, para instâncias com $p > 5$, as buscas locais implementadas na literatura são parciais e limitadas. Especificamente, se $p >> 5$, elas cobrem uma porção bem pequena da vizinhança de um vetor topologia. No entanto, o aumento do número de vizinhos considerados leva a um substancial aumento do tempo computacional da busca. Tal fato é decorrente de que a avaliação do custo de cada vetor topologia depende da execução de um método iterativo, onde cada iteração é realizada em $O(np)$ e o número de iterações pode ser muito maior que p.

As heurísticas de busca local podem ser implementadas visando obter minimos locais com o menor tempo computacional possível. Dessa forma, é adotada a estratégia *first improving*. Tal estratégia consite em substituir a solução corrente pela primeira solução de custo menor que o da solução corrente durante a busca. Por outro lado, a vizinhança corrente pode ser substituída pela solução de menor custo dentre todas as soluções pertencentes ao conjunto de soluções da vizinhança considerada, sendo esta estratégia denominada *best improving*. A busca gulosa efetuada nos algoritmos implementados usando *best improving* tem mostrado, experimentalmente, convergir prematuramente a mínimos locais com custo maior que os algoritmos implementados usando *first improving*. Além disso, são executadas em tempo computacional mais elevado. Dessa forma, escolhemos utilizar, na busca local em vetores topologia, a estratégia *first improving*.

Observando a monotonicidade no decrescimento do tamanho da árvore a cada iteração do procedimento de minimização de árvores, notamos que se o valor do erro da árvore 2.19 for muito menor que a diferença entre o seu tamanho , em uma dada iteração, e o tamanho da menor árvore encontrada pela busca local, então nas iterações seguintes a árvore não reduzirá o seu tamanho a ponto de ser menor que o da menor árvore encontrada até o momento. Dessa forma, a minimização desta árvore pode ser interrompida antes da convergência do método iterativo ser atingida.

Para analisar os efeitos da utilização desse critério de parada na minimização das topologias de Steiner consideradas na busca local, realizamos alguns testes com-

putacionais com implementações da busca local em vetores topologia original e a sua versão com interrupção, que chamaremos de acelerada. Aplicamos ambas à instâncias com 50 pontos dados, com as mesmas soluções iniciais aleatórias e com o tamanho da vizinhança igual a $5p$.

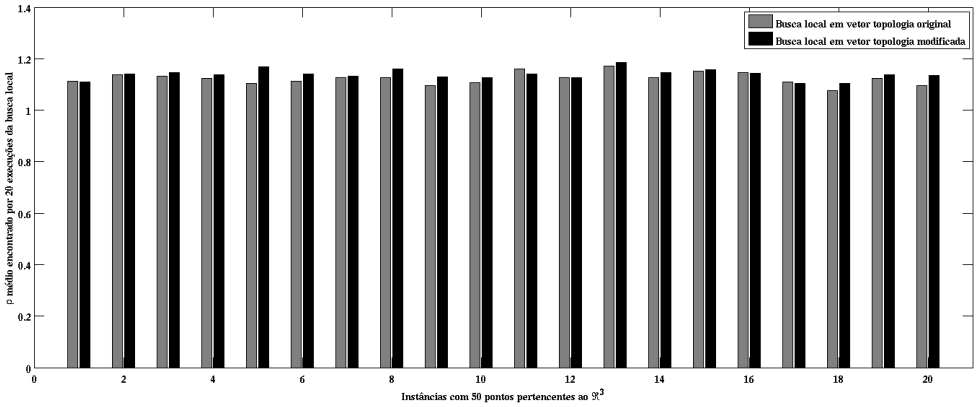

Figura 3.5: Comparação dos valores de ρ encontrados entre a busca local em vetores topologia original e a versão acelerada

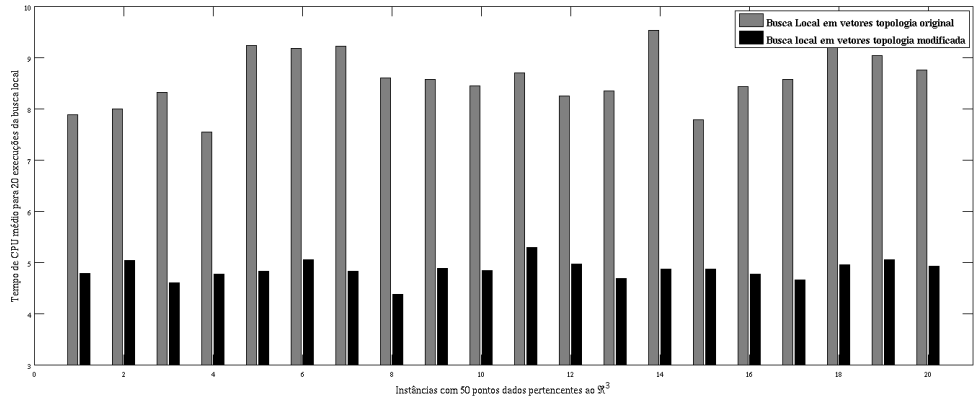

Figura 3.6: Comparação dos tempos de CPU obtidos pela aplicação das duas buscas locais

Através dos resultados apresentados na figura 3.5, podemos verificar que a versão acelerada da busca local em vetores topologia encontra valores de ρ compatíveis com os obtidos pela versão original da heurística. Esta pequena diferença pode ser atribuída à aleatoriedade em que os movimentos são realizados, pois dificilmente as duas heurísticas efetuam o mesmo conjunto de movimentos na mesma ordem. Dessa forma, é natural a pequena diferença entre os valores obtidos pelas duas versões.

Por outro lado, observando a figura 3.6, podemos concluir que a versão modificada é executada utilizando um tempo de CPU bem menor, quando comparado com a busca original.

Com a finalidade de estabelecer uma melhor comparação do desempenho das duas versões da busca local, realizamos um segundo teste computacional utilizando as mesmas instâncias e soluções iniciais geradas para o primeiro teste. Porém, no novo teste realizamos uma busca completa, ou seja, consideramos toda a vizinhança do vetor topologia corrente durante a busca local.

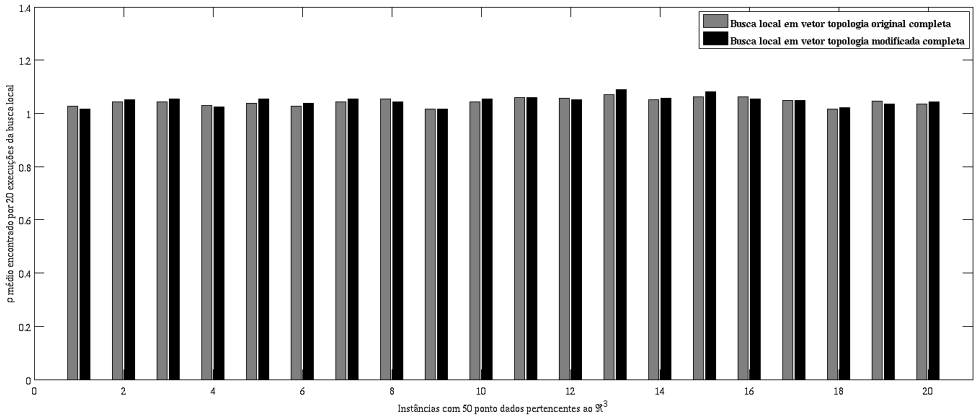

Figura 3.7: Comparação dos valores de ρ encontrados entre a busca local em vetores topologia original e a versão acelerada, ambas considerando toda a vizinhança.

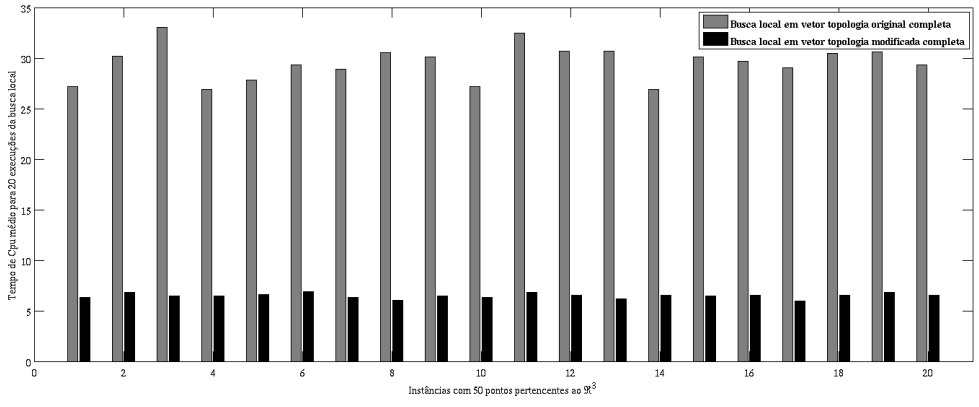

Figura 3.8: Comparação dos tempos de CPU obtidos pela aplicação das duas buscas locais completas

Para o segundo teste, com base nos resultados apresentados na figura 3.7, verificamos que as árvores mínimas relativas obtidas pelas duas versões da heurística

de busca local retornaram valores relativamente mais próximos que os apresentados no primeiro teste, onde foi considerada apenas uma pequena porção da vizinhança durante a busca.

Com relação ao tempo de CPU, para o segundo teste, podemos observar, através da figura 3.8, que o tempo de CPU exigido pela busca local completa em vetor topologia acelerada foi bem inferior ao tempo de CPU exigido pela sua versão original.

Observe-se que a heurística de busca local descrita acima realiza a busca no espaço das árvores mínimas relativas, já que não há garantia de que os vetores topologia examinados durante o procedimento estarão relacionados a uma árvore de Steiner - o que seria ideal, uma vez que a árvore mínima de Steiner é a menor dentre todas as arvores de Steiner.

No entanto, sabemos que cada vetor topologia está relacionado a uma árvore com topologia de Steiner cheia e que, após a minimização, a árvore mínima relativa obtida não será uma árvore de Steiner, se esta tiver uma ou mais ligações entre pares de pontos de Steiner com tamanho nulo (ver seção 2.2 do capítulo 2 desta tese). Dessa forma, para garantir que estaremos mantendo a busca no espaço das árvores de Steiner, inserimos um movimento, além do utilizado nesta busca local, que será realizado toda vez que a busca local se deparar com um vetor topologia cuja árvore mínima relativa associada não for uma árvore de Steiner.

Esse movimento adicional consiste em realizar a troca entre duas ligações,(a, i) e (b, j), não nulas e adjacentes a dois pontos de Steiner, i e j , sempre que a ligação (i, j) que os conecta for nula. Esse movimento é equivalente ao realizado no processo de interação contido na heurística de Relaxação Dinâmica Estendida, sendo também utilizado sempre que dois de pontos de Steiner estiverem conectados por uma ligação nula. Apoś aplicarmos este movimento a todos os pares de pontos de Steiner degenerados, em uma árvore mínima relativa com topologia de Steiner cheia, podemos garantir que a árvore resultante será uma árvore de Steiner de menor tamanho que a original [22].

E toda vez que a busca local se deparar com um vetor topologia cuja árvore mínima relativa possua um ou mais pares de pontos de Steiner degenerados, um movimento adicional é realizado, até que se encontre uma árvore de Steiner. Se o

tamanho dessa árvore de Steiner for menor que o tamanho da menor árvore encontrada pela busca local até a presente iteração, então o vetor topologia relacionado a sua topologia é considerado como a solução corrente. E a busca por uma árvore de Steiner de menor tamanho continua na vizinhança deste novo vetor topologia.

Para avaliar o comportamento da busca local utilizando esse movimento adicional, acrescentamos este movimento à busca local com aceleração proposta anteriormente e aplicamos ao mesmo conjunto de teste anterior. Também consideramos a vizinhança completa. As comparações com a proposta anterior (apenas com aceleração e vizinhança completa) estão nas figuras 3.9 e 3.10.

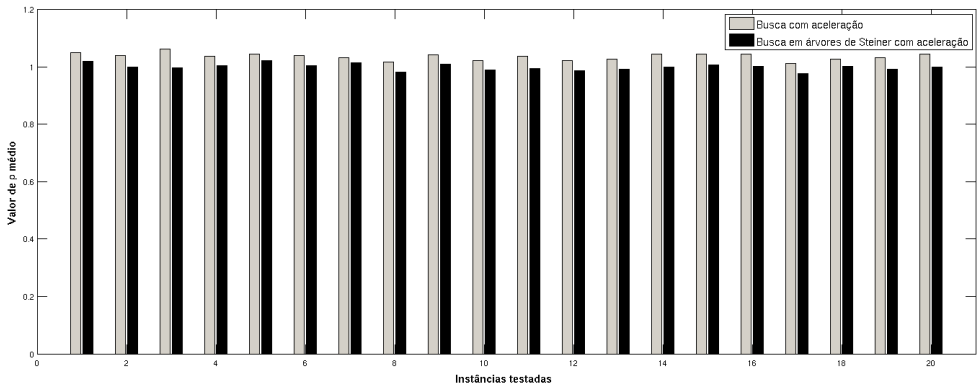

Figura 3.9: Comparação entre os valores de ρ encontrados pela busca local apenas com aceleração e pela versão acrescida do movimento adicional (busca em árvores de Steiner), ambas considerando toda a vizinhança.

Através dos gráficos 3.9 e 3.10 podemos constatar que a versão da busca local árvores de Steiner encontra mínimos locais para o PSE com valor de ρ menor que o obtido pela versão anterior, utilizando um tempo de CPU bem menor para encontrá-los.

Com base nestes testes, escolhemos a versão modificada (com aceleração e movimento adicional) da busca local em vetor topologia como procedimento de busca local, sendo também escolhida a busca em vizinhança completa por apresentar uma maior redução do valor de ρ, com um pequeno acréscimo no tempo de CPU.

A seguir, apresentamos o algoritmo do procedimento de busca local baseado em vetores topologia.

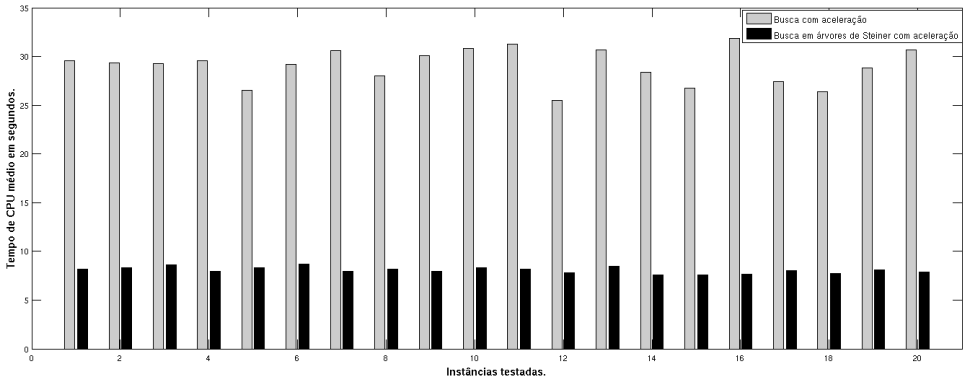

Figura 3.10: Comparação entre os tempos de CPU obtidos pela busca local apenas com aceleração e pela versão acrescida do movimento adicional (busca em árvores de Steiner), ambas considerando toda a vizinhança.

Perturbação

O procedimento de perturbação é responsável pela obtenção de um vetor topologia v', a partir de um mínimo local v retornado pela busca local, esperando que este se encontre na região de atração de um mínimo local diferente de v.

A perturbação é realizada escolhendo-se aleatoriamente k componentes entre as p-3 componentes do vetor topologia v, formando o conjunto $\{\pi_1, \pi_2, ..., \pi_k\}$ das componentes escolhidas, e fazendo $v' \leftarrow v$. Em seguida, para todo componente π_i deste conjunto, um valor $v'(\pi_i)$ inteiro é escolhido no intervalo $[1, 2\pi_i + 1]$, tal que $v'(\pi_i) \neq v(\pi_i)$. Após realizar estas k operações o vetor v' resultante será uma perturbação do vetor topologia v.

É importante ressaltar que a escolha do parâmetro k do procedimento de perturbação é de especial importância, pois determina a intensidade com que a perturbação é realizada. Desta forma, se k for muito pequeno é possível que a solução perturbada continue pertencendo à região de atração do mínimo local, sobre o qual foi realizada a perturbação[43]. Nesse caso, torna-se muito difícil encontrar um outro mínimo local com custo menor que o da atual melhor solução. No entanto, se k for muito grande, a mataheurística ILS terá o comportamento da metaheurística de Reinicialização Aleatória[50], que por sua vez, à medida que o tamanho das instâncias aumenta, tem reduzida sua probabilidade de encontrar um mínimo local com custo menor.

Entrada: v - Vetor topologia e ρ - Razão de Steiner para v;

para $j = 1$ *até máximo_vizinhos* **faça**

$v' \leftarrow v$;

$i \leftarrow \text{rand}(1, p-3)$;`/* rand(a,b) gera um número pseudo-aleatório entre a e b */`;

$v'(i) \leftarrow \text{rand}(1, 2i+1)$;

$(\rho', \text{Árvore}) \leftarrow \text{Minimiza}(v')$; `/* Minimiza(a) realiza a minimização da árvore associada ao vetor topologia a */`;

para cada (i,j) *:* $(i,j) \in v'$, $i,j \in S$ *e* $\|x_i - x_j\| \approx 0$ **faça**

Troca(v'); ; `/* Troca(a) efetua uma troca de arestas entre pontos de Steiner degenerados */`

;

$(\rho', \text{Árvore}) \leftarrow \text{Minimiza}(v')$;

fim

se $\rho' < \rho$ **então**

$j \leftarrow 1$;

$\rho \leftarrow \rho'$;

$v \leftarrow v'$;

fim

fim

Algoritmo 2: Busca Local em Vetores Topologia

No algoritmo3 é apresentado o procedimento de perturbação para o PSE.

Entrada: v- Vetor Topologia e k - Tamanho da perturbação

para $i = 1$ *até* k **faça**

$h(i) \leftarrow \text{rand}(1, p-3)$; `/*` $h(i) \neq h(j)\ \forall i,j \in \{1,...,k\}$ `*/`;

$v'(h(i)) \leftarrow \text{rand}(1, 2(h(i))+1)$; `/*` $v(h(i)) \neq v'(h(i))$ `*/`;

$v(h(i)) \leftarrow v'(h(i))$;

fim

Algoritmo 3: Procedimento de perturbação

Aceitação

O procedimento de aceitação é o responsável por efetivar a transição de uma vizinhança da busca local para outra vizinhaça encontrada através da aplicação do procedimento de perturbação a um mínimo local. O procedimento de busca local quando aplicado a essa vizinhança pode conduzir a busca a um mínimo local com custo menor do que o da melhor solução encontrada atualmente.

Com isso, o critério de aceitação utilizado neste procedimento determina quando a metaheurística de busca local deve ser intensificada em uma região ou diversificada para outra região de busca. Assim, a escolha apropriada desse critério contribui para que se encontre uma trajetória eficiente em V^* (conjunto de mínimos locais) que leve a um mínimo local de menor custo.

Diferentes critérios de aceitação podem ser aplicados para aceitar o vetor $v^{*'}$ como nova solução corrente da busca local.

Podemos, por exemplo, utilizar um critério de intensificação, o qual aceita somente soluções que sejam melhores que a melhor solução encontrada até o momento pela metaheurística:

$$\text{Melhor}(v^*, v^{*'}) = \begin{cases} v^{*'}, \text{ se } f(v^{*'}) < f(v^*) \\ v^*, \text{ caso contrário} \end{cases} . \quad (3.5)$$

Alternativamente, podemos utilizar um critério que aceite qualquer vetor topologia $v^{*'}$, levando a metaheurística ILS a realizar um caminho aleatório (CA) no conjunto de soluções viáveis para o problema:

$$CA(v^*, v^{*'}) = v^{*'} . \quad (3.6)$$

Os dois critérios mencionados acima representam critérios de aceitação extremos, onde o primeiro visa somente a intensificação da região de busca e o segundo a sua diversificação. Ademais, a adoção de um dos dois critérios pode impedir que mínimos locais mais próximos do mínimo global sejam encontrados.

Nos algoritmos implementados neste trabalho usaremos três critérios de aceitação, onde um destes critérios escolhidos é o critério 3.5, devido a sua extensa utilização em implementações da metaheurística ILS encontradas na literatura. Os outros dois critérios são um meio termo entre os dois critérios extremos, ora realizando uma intensificação, ora uma diversificação, à medida que forem necessários.

Assim, escolhemos um critério Markoviano, tal como utilizado por Martin et al. em [44] no algoritmo *Large-step Markov chains*, e um segundo critério que utiliza informações obtidas de um conjunto elite de soluções[35].

O critério de aceitação Markoviano é semelhante ao utilizado na metaheurística *Simulated Annealing*[33]. Por esse critério, se o vetor topologia $v^{*'}$ tem custo $f(v^{*'})$ menor que o custo $f(v^*)$ do vetor topologia v^*, então o vetor topologia $v^{*'}$ é aceito. Caso $f(v^*) \leq f(v^{*'})$ o vetor topologia $v^{*'}$ é aceito com probabilidade $\exp \frac{(f(v^*)-f(v^{*'}))}{T}$, onde T, um parâmetro chamado temperatura[33], é reduzido a cada iteração da metaheurística ILS. Dessa forma, iniciando com uma temperatura alta, o critério diversifica a busca local e, ao final, com a temperatura reduzida, o critério intensifica a busca local. No algoritmo 4, podemos ver as etapas referente ao procedimento de aceitação utilizando este critério.

Entrada: ρ - Valor da razão de Steiner para o melhor vetor topologia encontrado , ρ' - Valor da razão de Steiner para o vetor topologia candidato a aceitação e T - Temperatura

se $\rho' \leq \rho$ **então**

    Aceita o vetor topologia com valor da razão de Steiner ρ';

senão

    $r = \text{rand}(0, 1)$; `/* gerar um número real entre 0 e 1 */`;

    se $r \leq \exp \frac{(\rho-\rho')}{T}$ **então**

        Aceita o vetor topologia com valor da razão de Steiner ρ';

    fim

fim

Algoritmo 4: Procedimento de aceitação Markoviano

Para o terceiro critério de aceitação foi criado um conjunto E denominado Elite, o qual armazena um número pré-especificado de vetores topologia, associados às árvores de Steiner de menor tamanho encontradas até o momento. Com base neste conjunto, são calculadas a média, μ, e o desvio padrão, σ, considerando o custo de cada vetor topologia pertencente ao conjunto E. Utilizando estes valores, o critério aceita um vetor topologia $v^{*'}$ se o seu custo $f(v^{*'})$ pertencer ao intervalo $[\mu - 2\sigma, \mu + 2\sigma]$ ou se $f(v^{*'}) < f(v^*)$. Se um vetor topologia é aceito, então o conjunto E é

atualizado, substituindo-se o vetor topologia com maior custo pelo vetor topologia aceito. Devendo os valores μ e σ serem, portanto, recalculados em uma próxima chamada do procedimento. O Algoritmo 5 descreve o procedimento de aceitação associado a este critério.

Entrada: ρ - Valor da razão de Steiner para o melhor vetor topologia encontrado , ρ' - Valor da razão de Steiner para o vetor topologia candidato a aceitação e E - Conjunto elite de soluções

se $\rho' \leq \rho$ **então**

    Aceita o vetor topologia com valor da razão de Steiner ρ';

senão

    $\mu \leftarrow$ Média(E);

    $\sigma \leftarrow$ Desvio_Padrão(E);

    se $\rho' \in [\mu - 2\sigma, \mu + 2\sigma]$ **então**

        Aceita o vetor topologia com valor da razão de Steiner ρ';

        Atualiza(E);

    fim

fim

Algoritmo 5: Procedimento de aceitação com conjunto elite

Capítulo 4

Resultados Computacionais

O presente capítulo apresenta um conjunto de resultados computacionais obtidos mediante a aplicação de um algoritmo de Relaxação Dinâmica melhorado e de seis algoritmos ILS. A partir desses resultados, foi possível avaliar a robustez e a eficiência dos algoritmos.

4.1 Algoritmos Implementados

Para a proposta de melhoria na heurística de Relaxação Dinâmica Estendida descrita na seção 3.1 do capítulo 3, foram implementados o algoritmo de Relaxação Dinâmica Estendida original (RE), e a sua versão com a aplicação da metaheurística Multi-Start (REM). Também foram implementados os dois algoritmos anteriores acrescentandos da modificação proposta, sendo denominados RE mod. e REM mod. respectivamente.

No que concerne a aplicação da metaheurística ILS ao PSE, foram implementados seis algoritmos. Tais algoritmos são o resultado da combinação de dois procedimentos de inicialização, três procedimentos de aceitação, um procedimento de busca local e um procedimento de perturbação.

Na tabela abaixo mostramos as combinações consideradas para a implementação dos algoritmos, sendo possível observar quais variantes de cada procedimento foram usadas em cada um dos algoritmos.

Algoritmos	ILS1	ILS2	ILS3	ILS4	ILS5	ILS6
Inicialização	Inserção de pontos de Steiner na AGM	Inserção de pontos de Steiner na AGM	Inserção de pontos de Steiner na AGM	Construção gulosa	Construção gulosa	Construção gulosa
Aceitação	Melhor	Conjunto elite	Markoviano	Melhor	Conjunto elite	Markoviano

Seguindo o padrão de outros trabalhos, escolhemos como critério de parada para estes algoritmos ILS o número de iterações sem a obtenção de um vetor topologia que produza uma redução no valor de ρ, em relação ao valor atual. Para instâncias com até 11 pontos dados foram consideradas no máximo 10 iterações e para as instâncias maiores foram consideradas no máximo 50 iterações.

Para os algoritmos baseados na heurística de Relaxação Dinâmica Estendida, o critério de parada utilizado e a atualização dos valores de δ e η foram os mesmos usados em [42]. Assim, o valor de η é reduzido em $\frac{1}{5}$ de seu valor inicial a cada vinte iterações a partir da iteração 100, assumindo valor zero, $\eta = 0$, na iteração 180, enquanto que o valor de δ permanece constante até a iteração 200 e, a partir desta, o seu valor é reduzido pela metade a cada 20 iterações, até a iteração 400, quando o algoritmo é terminado.

As árvores geradoras mínimas obtidas pelos algoritmos implementados neste trabalho foram construídas por uma versão do algoritmo de Prim[51] contida, como uma subrotina, no código do algoritmo de Smith apresentado em [9].

Todos os algoritmos foram implementados em linguagem C padrão e compilados usando o compilador GCC 4.2. Os testes computacionais foram realizados em ambiente Linux, com sistema operacional Ubuntu 8.04, em um computador dotado de processador Intel Xeon Quad-core 3.0 Ghz com 16 Gb de memória RAM.

4.2 Conjunto de Teste

Para efeitos de comparação, utilizamos instâncias com 8, 9, 10, 11, 50, 100 e 250 pontos pertencentes ao espaço tridimensional.Essas mesmas instâncias foram consideradas por [18] para comparar o algoritmo GRASP aplicado ao PSE às demais heurísticas.

Para cada número de pontos acima, foram geradas aleatoriamente 1000 instâncias, considerando uma distribuição uniforme em um cubo ou hiper-cubo unitário. No particular caso de 50, 100 e 250 pontos, foram geradas 15 instâncias de cada conjunto.O número reduzido de instâncias deve-se ao elevado tempo computacional necessário para resolvê-las.

Considerando o mesmo mecanismo de geração descrito acima, também foram

geradas para este trabalho 1000 instâncias com 10 pontos pertencentes ao $\Re^4$ e ao $\Re^5$.

Para os algoritmos baseados na Relaxação Dinâmica Estendida, foram considerados conjuntos de 1000, 10000 e 100000 pontos pertencentes ao $\Re^3$. Para cada um destes conjuntos, foram geradas 100 instâncias aleatoriamente em um cubo unitário.

4.3 Ajuste dos Parâmetros

É fato conhecido que uma boa performance dos algoritmos baseados em metaheurísticas está diretamente associada a um ajuste adequado de seus parâmetros de entrada. Para os algoritmos ILS implementados neste trabalho, os parâmetros considerados são os seguintes: (i) o tamanho da perturbação no procedimento de perturbação; (ii) o número de vetores topologia armazenados no conjunto elite no procedimento de aceitação com conjunto elite; e (iii) o valor da temperatura no procedimento de aceitação Markoviano. No caso dos algoritmos de Relaxação Dinâmica Estendida, temos: (i) o valor inicial δ no procedimento de evolução; (ii) o valor inicial para η no procedimento de interação; e (iii) o número de repetições para a versão Multi-Start.

No entanto, observa-se que não existe uma regra geral a ser aplicada para o ajuste desses parâmetros. Desta forma, realizamos previamente alguns experimentos computacionais, com o objetivo de encontrar um bom conjunto de parâmetros.

Para a definição da quantidade de vetores topologia e da temperatura nos procedimentos de aceitação, realizamos 100 execuções dos algorimos ILS2, ILS3, ILS5 e ILS6. Nessas execuções, foi considerado um subconjunto contendo aproximadamente 10 % das instâncias do conjunto total de instâncias.

A partir desses testes, notamos que a qualidade das soluções não aumentava quando considerávamos o número de vetores topologia no conjunto elite maior que 5 e a temperatura maior que 2. Dessa forma, após esses experimentos prévios, fixamos os parâmetros com esses valores.

Já para o tamanho da perturbação, observou-se que a performace de todos os algoritmos ILS implementados apresentou uma elevada variação quanto ao número de pontos dados em cada uma das instâncias e ao procedimento de inicialização

utilizado pelo algoritmo. Com isso, resolvemos conduzir um experimento mais detalhado para a determinação deste parâmetro.

Para avaliar o impacto de cada perturbação nos algoritmos considerados, implementamos uma versão simplificada de um algoritmo ILS. Nessa versão é realizada uma busca local em uma solução inicial obtida por um dos procedimentos de inicialização, produzindo uma solução v. Em seguida, aplica-se uma perturbação de tamanho k em tal solução e produz-se uma nova solução v'. Essa solução é utilizada como ponto de partida por um procedimento de busca local, que produz um vetor topologia v'^{*}. Caso haja redução no valor da solução, ou seja, se $\rho_{v'*} \leq \rho_v$, o algoritmo é finalizado. Caso contrário, um novo vetor topologia v' é encontrado pela perturbação de tamanho k do vetor v, e sobre este é novamente aplicada a busca local. Esses dois procedimentos são realizados até que uma solução $\rho_{v'*} \leq \rho_v$ seja encontrada ou até um limite de tempo de CPU seja atingido.

Entrada: k - Tamanho da perturbação e instância com p pontos
$v_0 \leftarrow$ Procedimento-de-inicialização(instância);
$(v, \rho_v) \leftarrow$ Busca-local(v_0);
repita
    $v' \leftarrow$ Perturbação(v, k);
    $v'* \leftarrow$ Busca-local(v');
até $\rho_v \geq \rho_{v'*}$;

Algoritmo 6: Algoritmo ILS reduzido.

Para cada instância, foi selecionado um conjunto de valores inteiros no intervalo [1,p-3] para serem testados como valores de perturbação pelo algoritmo ILS simplificado.

Considerando uma instância para cada quantidade de pontos dados no conjunto de teste, esse experimento consistiu em 100 execuções de um algoritmo ILS simplificado para cada instância, para cada algoritmo de inicialização e para cada tamanho de perturbação selecionado. O algoritmo reduzido avalia o impacto de cada tamanho de perturbação na obtenção de uma solução melhor que a solução inicial em cada classe de instâncias e para os dois procedimentos de inicialização.

Com os valores da diferença entre o valor de ρ para a solução inicial e o valor

de ρ para a solução final para as 100 execuções e o número de iterações necessárias para a obtenção dessa solução, construímos três histogramas, quais sejam: um para o valor das diferenças, um para o número de iterações e outro para a razão entre a diferença e o número de iterações. Observando os histogramas, conseguimos analisar qual tamanho de perturbação possibilita a mudança de um mínimo local para outro de melhor qualidade. Na figura 4.1 encontra-se o conjunto de histogramas para as instâncias com 50 pontos. Os histogramas para as demais classes de instâncias encontram-se no apêndice deste trabalho.

A partir deste experimento, foi possível definir o valor do parâmetro associado ao tamanho da perturbação. Tal parâmetro é utilizado tanto no procedimento de construção guloso, quanto no procedimento de inserção de pontos. Na tabela 4.3 abaixo temos o valor definido para este parâmetro, considerando cada um dos procedimentos e cada um dos conjuntos de pontos.

Como resultado do experimento, estimamos um tamanho de perturbação para cada classe de tamanho de instâncias e para cada procedimento de inicialização considerado. Apresentamos esses valores na tabela 4.3.

Conjuntos de Pontos	8	9	10	11	50	100	250
Inserção de pontos	3	3	3	3	1	1	1
Guloso	3	3	3	3	10	15	15

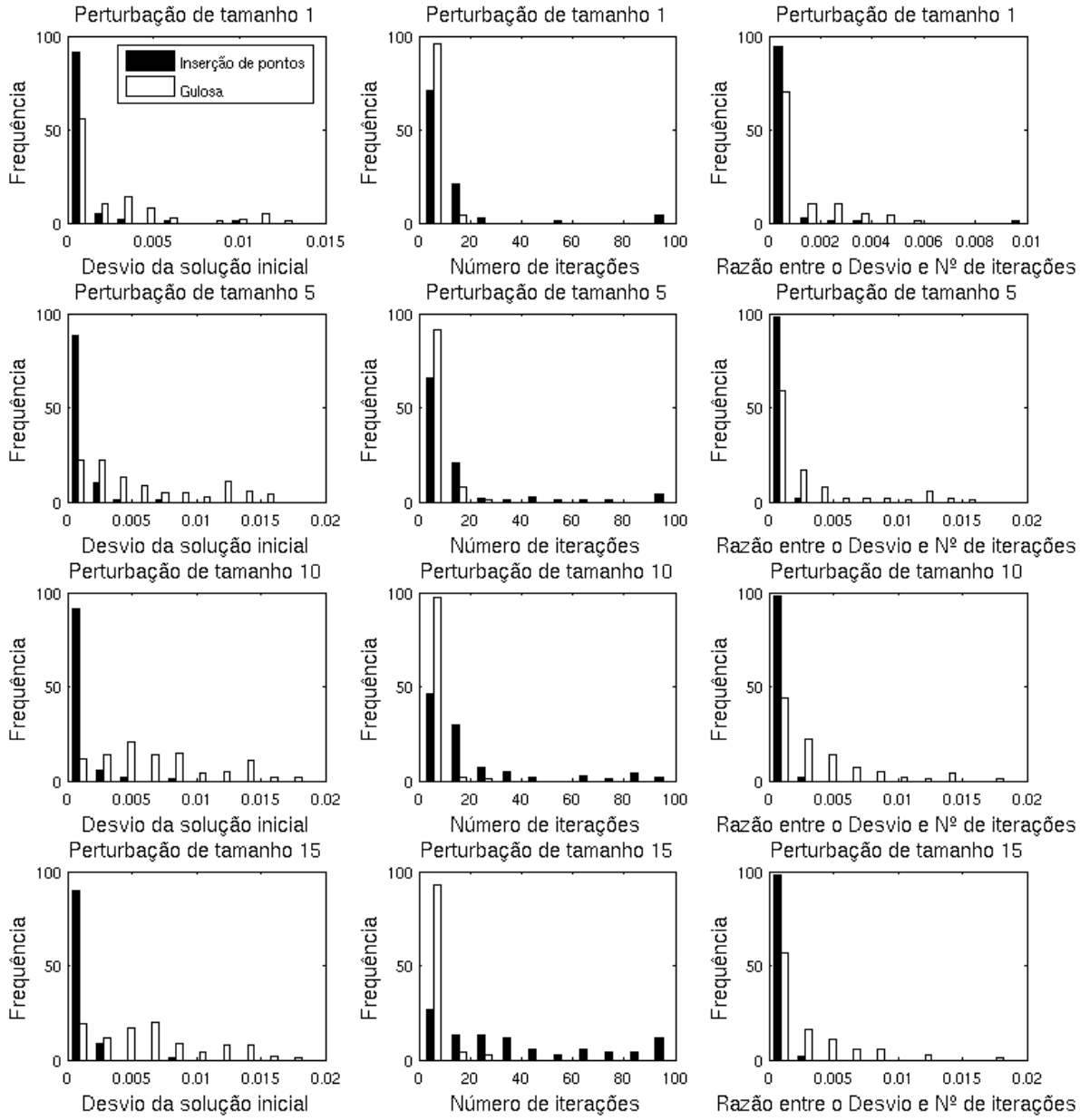

Figura 4.1: Histogramas para o ajuste do tamanho da perturbação para instâncias com 50 pontos dados.

Para o ajuste dos parâmetros (σ,η) utilizados pelos algoritmos de Relaxação Dinâmica Estendida, realizou-se um experimeto que consistiu em executar várias vezes os algoritmos com os parâmetros δ e η assumindo valores entre 0.01 e 1, com um incremento de 0.01.

Como resultado deste experimento, pode-se concluir que os valores de δ e η que produziram em média soluções melhores e mais estáveis seriam 0.12 e 0.18 respectivamente.

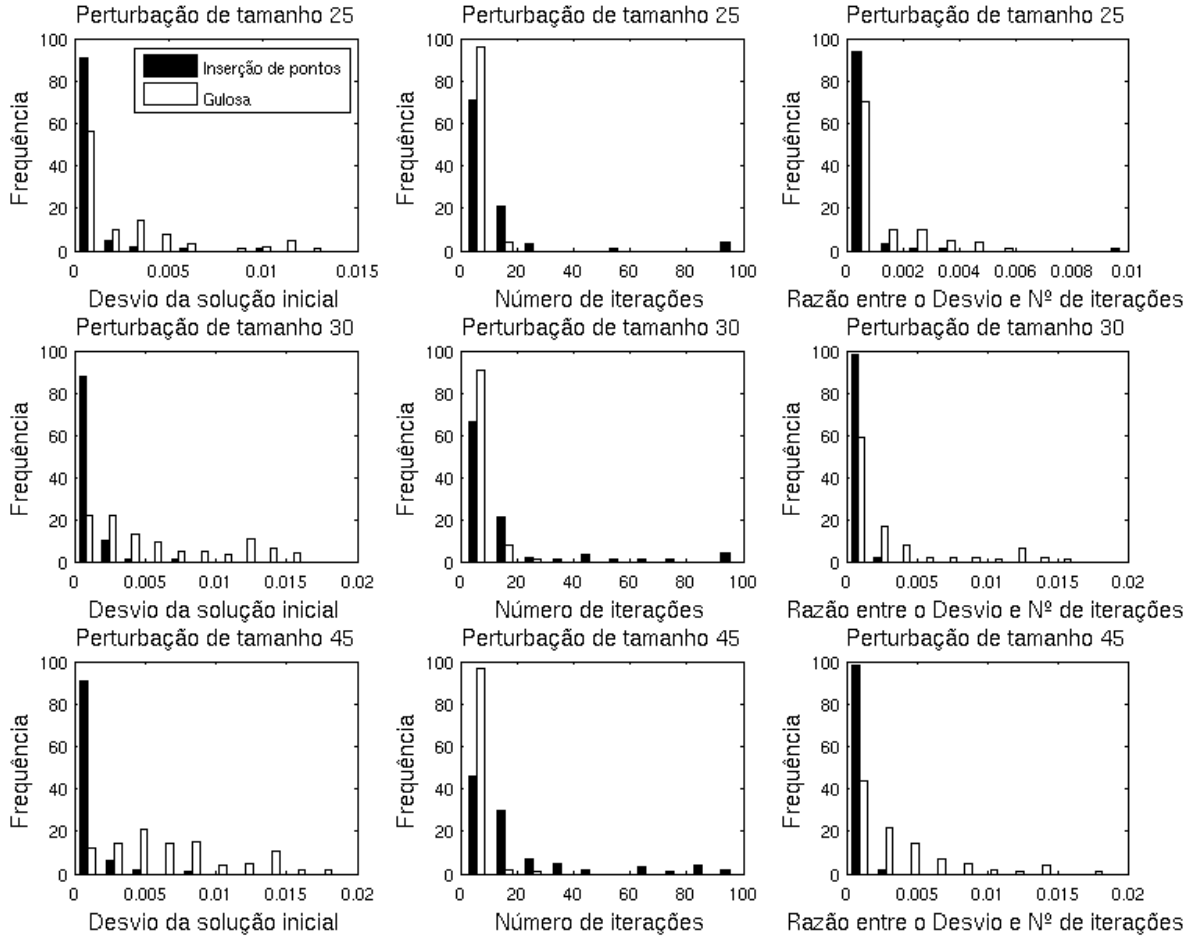

Figura 4.2: continuação da figura 4.1 - Histogramas para o ajuste do tamanho da perturbação para instâncias com 50 pontos dados.

4.4 Resultados Obtidos

Para relatar com o máximo de clareza possível os resultados encontrados com o experimento computacional, apresentaremos os resultados para cada conjunto de instâncias separadamente. Começando com as instâncias de pequeno porte, contendo de 8 a 11 pontos dados, depois para instâncias de médio porte, com 50 a 250 pontos dados, e finalmente instâncias de grande porte, com 1000 a 100000 pontos dados.

Foram realizadas 10 repetições do experimento computacional, com o objetivo de verificar a estabilidade e a robustez dos algoritmos propostos. Os valores reportados correspondem à iteração onde o experimento computacional produziu os melhores resultados.

Além de efetuar a comparação entre os algoritmos propostos neste trabalho,

comparamos também os resultados destes com os obtidos pelos algoritmos Microcanônico[10],[38] e GRASP[18]. Tais algoritmos são os mais recentes encontrados na literatura para encontrar soluções para o PSE em $\Re^n$. O código fonte em linguagem C para o algoritmo Microcanônico foi cedido pelos autores do artigo [10]. Já no caso do GRASP, a comparação dos resultados ficou limitada aos resultados reportados em [18], encontrados para o mesmo conjunto de instâncias de pequeno e médio porte utilizados neste experimento.

Tendo em vista que os experimentos computacionais apresentados neste trabalho e no trabalho de [18] foram realizados em computadores diferentes, foi realizado um escalamento do tempo de CPU, de forma a minimizar as discrepâncias com relação aos tempos de CPU dos computadores. Este escalamento consiste em multiplicar o tempo de CPU obtido pelo algoritmo GRASP pela razão entre o clock do processador do computador utilizado para obter os resultados em [18] e o clock do processador do computador utilizado nos experimentos computacionais desta dissertação.

4.4.1 Instâncias de Pequeno Porte

A partir das tabelas apresentadas a seguir, avaliou-se a qualidade dos resultados encontrados pelos algoritmos ILS, GRASP, Microcanônico (Micro O), Exato e de Relaxação para todas as instâncias de pequeno porte consideradas.

Nas tabelas que serão apresentadas a seguir, cada coluna está associada com um conjunto de instâncias.

Alguns dos valores que seriam reportados nas tabelas não puderam ser obtidos. No caso dos valores referentes ao algoritmo GRASP, a ausência deve-se a inexistência dos mesmos no trabalho de Rocha[18]. Já no caso dos algoritmos baseados na heurística de Relaxação Dinâmica Estendida, os valores ausentes são referentes à não obtenção de soluções intermediárias ao longo das iterações desses algoritmos. Nesses algoritmos, a única solução produzida é a apresentada após o critério de parada ser atingido. Nas tabelas, todos os valores que não puderam ser obtidos são representados por um traço.

Na tabela 4.1, são apresentados os valores do desvio padrão e da média calculados para os valores de ρ produzidos pelos algoritmos para cada conjunto de instâncias de pequeno porte.

Observando os valores da média e do desvio padrão reportados na tabela 4.1, notamos que para as instâncias com 8 e 9 pontos em $\Re^3$ os valores obtidos pelos algoritmos ILS são praticamente os mesmos que os obtidos pelo algoritmo exato. Para as instâncias com 10 e 11 pontos em $\Re^3$, os valores da média e desvio padrão, quando não são exatamente iguais ao do algoritmo exato, são muito próximos. O mesmo pode ser observado com relação aos valores encontrados para instâncias com 10 pontos em 4 e 5 dimensões.

Em relação aos algoritmos baseados na heurística de Relaxação Dinâmica Estendida, ambas as versões modificadas apresentaram a média do valor de ρ mais próxima da média obtida para os resultados encontrados com o algoritmo exato. Em particular, os resultados foram mais próximos quando utilizada a versão Multi-Start com a versão modificada.

Tabela 4.1: Média e desvio padrão dos valores de ρ obtidos por cada algoritmo aplicado nas instâncias de pequeno porte.

Algoritmos	Medidas	8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
Exato	Média	0.94676	0.94640	0.94676	0.94683	0.92778	0.91163
	Desvio padrão	0.01951	0.01775	0.01753	0.01584	0.01834	0.01869
ILS1	Média	0.94676	0.94640	0.94679	0.94685	0.92779	0.91165
	Desvio padrão	0.01951	0.01775	0.01753	0.01585	0.01834	0.01869
ILS2	Média	0.94676	0.94640	0.94676	0.94686	0.92780	0.91164
	Desvio padrão	0.01951	0.01776	0.01753	0.01586	0.01834	0.01869
ILS3	Média	0.94676	0.94640	0.94676	0.94685	0.92780	0.91166
	Desvio padrão	0.01951	0.01775	0.01753	0.01583	0.01836	0.01869
ILS4	Média	0.94676	0.94641	0.94676	0.94687	0.92781	0.91165
	Desvio padrão	0.01951	0.01777	0.01753	0.01585	0.01833	0.01871
ILS5	Média	0.94676	0.94641	0.94677	0.94685	0.92780	0.91165
	Desvio padrão	0.01951	0.01774	0.01754	0.01584	0.01834	0.01870
ILS6	Média	0.94676	0.94640	0.94677	0.94688	0.92779	0.91164
	Desvio padrão	0.01951	0.01775	0.01754	0.01586	0.01834	0.01869
GRASP	Média	0.94701	0.94738	0.94757	0.94810	-	-
	Desvio padrão	0.01956	0.01826	0.02136	0.01626	-	-
micro O	Média	0.94755	0.94770	0.94829	0.94862	0.92923	0.91301
	Desvio padrão	0.01963	0.01809	0.01792	0.01629	0.01890	0.01912
RE	Média	0.95032	0.95034	0.95119	0.95132	0.93268	0.91658

Continua na próxima página.

Tabela 4.1 – continuação da página anterior.

Algoritmos	Medidas	8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
	Desvio padrão	0.02024	0.01852	0.01817	0.01672	0.01891	0.01940
REM	Média	0.94777	0.94783	0.94850	0.94823	0.92962	0.91339
	Desvio padrão	0.01968	0.01804	0.01773	0.01590	0.01853	0.01921
RE Mod.	Média	0.95009	0.95032	0.95132	0.95125	0.93248	0.91672
	Desvio padrão	0.02020	0.01852	0.01826	0.01686	0.01884	0.01928
REM Mod.	Média	0.94774	0.94777	0.94830	0.94821	0.92957	0.91335
	Desvio padrão	0.01970	0.01794	0.01767	0.01590	0.01858	0.01919

A tabela 4.2 traz as medidas resumo associadas aos valores de ρ, obtidos através da aplicação dos algoritmos para instâncias de pequeno porte. Nesta tabela, cada coluna representa um conjunto de instâncias.

Uma análise destas medidas indica que a distribuição dos valores de ρ encontrados pelos algoritmos ILS propostos neste trabalho é bem próxima da distribuição ótima dos valores de ρ.

Tabela 4.2: Resumo dos valores de ρ obtidos por cada algoritmo aplicado a instâncias de pequeno porte.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
Exato	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95940	0.95772	0.94113	0.92381
	Mediana	0.94656	0.94704	0.94773	0.94690	0.92700	0.91146
	1º quartil	0.93341	0.93442	0.93524	0.93683	0.91471	0.89917
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
ILS1	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95951	0.95775	0.94113	0.92386
	Mediana	0.94656	0.94706	0.94776	0.94691	0.92700	0.91146
	1º quartil	0.93341	0.93442	0.93535	0.93683	0.91471	0.89917
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
ILS2	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95940	0.95775	0.94113	0.92381
	Mediana	0.94656	0.94704	0.94773	0.94691	0.92700	0.91146
	1º quartil	0.93341	0.93442	0.93524	0.93683	0.91471	0.89917

Continua na próxima página.

Tabela 4.2 – continuação da página anterior.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
ILS3	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95940	0.95772	0.94113	0.92386
	Mediana	0.94656	0.94704	0.94774	0.94691	0.92700	0.91148
	1º quartil	0.93341	0.93442	0.93524	0.93683	0.91471	0.89917
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
ILS4	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95951	0.95775	0.94113	0.92381
	Mediana	0.94656	0.94704	0.94773	0.94699	0.92706	0.91148
	1º quartil	0.93341	0.93442	0.93524	0.93683	0.91473	0.89920
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
ILS5	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95940	0.95772	0.94113	0.92386
	Mediana	0.94656	0.94704	0.94773	0.94690	0.92708	0.91148
	1º quartil	0.93341	0.93454	0.93524	0.93688	0.91471	0.89917
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
ILS6	Máximo	0.99999	0.99396	0.99202	0.99403	0.99133	0.97043
	3º quartil	0.95978	0.95818	0.95940	0.95772	0.94113	0.92386
	Mediana	0.94656	0.94704	0.94773	0.94691	0.92700	0.91146
	1º quartil	0.93341	0.93442	0.93524	0.93693	0.91471	0.89917
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
micro O	Máximo	0.99999	0.99645	0.99608	0.99597	0.99133	0.97043
	3º quartil	0.96052	0.95969	0.96108	0.95955	0.94283	0.92529
	Mediana	0.94766	0.94796	0.94935	0.94860	0.92824	0.91274
	1º quartil	0.93414	0.93552	0.93672	0.93786	0.91570	0.90059
	Mínimo	0.88490	0.88848	0.88818	0.89660	0.86670	0.85717
RE	Máximo	1.01545	1.00067	0.99624	1.01517	0.99135	0.97297
	3º quartil	0.96438	0.96331	0.96352	0.96286	0.94594	0.92955
	Mediana	0.95012	0.95141	0.95178	0.95126	0.93227	0.91626
	1º quartil	0.93667	0.93866	0.93971	0.94013	0.91892	0.90369
	Mínimo	0.88491	0.88908	0.88821	0.89776	0.86670	0.85926
REM	Máximo	1.00000	0.99647	0.99610	0.99433	0.99135	0.97297
	3º quartil	0.96085	0.96005	0.96099	0.95899	0.94315	0.92628
	Mediana	0.94773	0.94845	0.94958	0.94817	0.92887	0.91241
	1º quartil	0.93452	0.93554	0.93719	0.93789	0.91617	0.90082

Continua na próxima página.

Tabela 4.2 – continuação da página anterior.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
	Mínimo	0.88491	0.88907	0.88819	0.89775	0.86670	0.85718
RE Mod.	Máximo	1.01177	1.00067	1.01608	1.01513	0.99133	0.97297
	3º quartil	0.96376	0.96328	0.96386	0.96286	0.94617	0.92920
	Mediana	0.95040	0.95139	0.95184	0.95071	0.93186	0.91626
	1º quartil	0.93686	0.93863	0.93965	0.94006	0.91873	0.90414
	Mínimo	0.88490	0.88907	0.88818	0.89774	0.86670	0.85795
REM Mod.	Máximo	0.99999	0.99692	0.99608	0.99432	0.99133	0.97297
	3º quartil	0.96079	0.95984	0.96070	0.95898	0.94314	0.92613
	Mediana	0.94777	0.94808	0.94944	0.94814	0.92879	0.91270
	1º quartil	0.93429	0.93569	0.93693	0.93788	0.91595	0.90082
	Mínimo	0.88490	0.88907	0.88818	0.89774	0.86670	0.85717

Já na tabela 4.3, verificamos para quantas das instâncias de pequeno porte os algoritmos retornaram a solução ótima (NSEE) e para quantas o algoritmo encontrou a solução ótima na primeira iteração do algoritmo (NSEPI).

A partir dos valores apresentados nessa tabela, pode-se observar que todos os algoritmos ILS propostos produziram um número maior de soluções exatas do que os demais algoritmos. Em particular, a versão dois do algoritmo ILS (ILS2) foi a que produziu os melhores resultados. Além disso, observou-se que no caso dos algoritmos baseados no algoritmo de Relaxação Dinâmica Estendida, a versão modificada aumentou expressivamente a quantidade de soluções exatas encontradas.

Tabela 4.3: Número de soluções exatas encontradas no final (NSEE) e encontradas na primeira iteração (NSEPI) pelos algoritmos dentre todas as intâncias de pequeno porte.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
ILS1	NSEE	1000	999	995	992	993	989
	NSEPI	870	817	787	756	693	631
ILS2	NSEE	1000	999	997	994	994	993
	NSEPI	946	919	886	881	853	846

Continua na próxima página.

Tabela 4.3 – continuação da página anterior.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
ILS3	NSEE	1000	999	998	993	992	990
	NSEPI	872	811	784	766	692	638
ILS4	NSEE	1000	998	997	990	989	992
	NSEPI	836	731	686	606	616	575
ILS5	NSEE	999	997	995	990	990	992
	NSEPI	816	754	685	619	618	589
ILS6	NSEE	1000	998	996	990	996	988
	NSEPI	836	731	686	635	616	589
GRASP	NSEE	931	902	878	878	-	-
	NSEPI	-	-	-	-	-	-
micro O	NSEE	907	838	802	754	770	757
	NSEPI	845	733	635	518	548	525
RE	NSEE	31	7	0	1	48	144
	NSEPI	-	-	-	-	-	-
REM	NSEE	55	13	3	3	110	327
	NSEPI	-	-	-	-	-	-
RE Mod.	NSEE	634	539	461	431	401	348
	NSEPI	-	-	-	-	-	-
REM Mod.	NSEE	881	819	801	784	737	727
	NSEPI	-	-	-	-	-	-

A tabela 4.4 apresenta o tempo médio total de CPU consumido por cada um dos algoritmos até que o seu critério de parada fosse satisfeito e o tempo médio de CPU consumido para encontrar a melhor solução.

Observamos que o tempo médio total de CPU consumido pelos algoritmos ILS foi menor que o consumido pelos outros algoritmos, não chegando a um segundo. Dentre os algoritmos ILS, vale destacar que mesmo o algoritmo ILS2 necessitando de um tempo médio total de CPU maior que os demais algoritmos ILS, este apresentou um tempo médio de CPU para encontrar a melhor solução aproximadamente igual aos demais algoritmos ILS.

No caso dos algoritmos baseados na Relaxação Dinâmica Estendida, podemos verificar que a modificação proposta não aumentou substancialmente o tempo de CPU necessário para se encontrar as soluções reportadas.

Tabela 4.4: Tempo médio de CPU em segundos gastos no total e para encontrar a melhor solução reportada para instâncias de pequeno porte.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
Exato	Tempo Médio Total	0.01143	0.06360	0.34137	1.92860	0.92603	1.96280
ILS1	Tempo Médio Total	0.07468	0.12624	0.19086	0.28190	0.18533	0.17221
	Tempo Médio Melhor Solução	0.00267	0.00603	0.01204	0.02056	0.01691	0.01808
ILS2	Tempo Médio Total	0.09367	0.15856	0.24025	0.35081	0.22909	0.20811
	Tempo Médio Melhor Solução	0.00230	0.00611	0.01255	0.02091	0.01625	0.01414
ILS3	Tempo Médio Total	0.07493	0.12808	0.19384	0.28440	0.18526	0.17326
	Tempo Médio Melhor Solução	0.00247	0.00741	0.01269	0.02165	0.01645	0.01776
ILS4	Tempo Médio Total	0.08249	0.14117	0.21588	0.32356	0.20812	0.19812
	Tempo Médio Melhor Solução	0.00424	0.01213	0.02361	0.04707	0.02733	0.03056
ILS5	Tempo Médio Total	0.08219	0.14106	0.21678	0.32247	0.20824	0.19717
	Tempo Médio Melhor Solução	0.00444	0.01146	0.02387	0.04371	0.02792	0.02969
ILS6	Tempo Médio Total	0.08262	0.14196	0.21721	0.32346	0.20949	0.19940
	Tempo Médio Melhor Solução	0.00414	0.01239	0.02346	0.04233	0.02799	0.03002
GRASP	Tempo Médio Total	0.84250	1.04720	1.30140	1.49200	-	-
	Tempo Médio Melhor Solução	-	-	-	-	-	-
micro O	Tempo Médio Total	0.15613	0.21458	0.27510	0.35066	0.28916	0.30039
	Tempo Médio Melhor Solução	0.00874	0.01723	0.03421	0.06490	0.04613	0.05070
RE	Tempo Médio Total	0.00404	0.00400	0.00400	0.00400	0.00400	0.00400
	Tempo Médio Melhor Solução	-	-	-	-	-	-
REM	Tempo Médio Total	0.03806	0.04540	0.05142	0.05828	0.05975	0.06943
	Tempo Médio Melhor Solução	-	-	-	-	-	-
RE Mod.	Tempo Médio Total	0.00408	0.00405	0.00406	0.00404	0.00406	0.00404
	Tempo Médio Melhor Solução	-	-	-	-	-	-
REM Mod.	Tempo Médio Total	0.04278	0.04757	0.07020	0.06577	0.06736	0.07416
	Tempo Médio Melhor Solução	-	-	-	-	-	-

Na tabela 4.5 são apresentados o desvio padrão e a média do percentual da diferença entre os valores de ρ para a melhor solução encontrada e a segunda melhor solução. Também é apresentado o tempo médio de CPU gasto para encontrar a segunda melhor solução. Com estas informações, é possível analisar o tempo necessário para encontrar a solução mais próxima da solução retornada pelos algoritmos e qual

foi o esforço dispendido (tempo de CPU) para encontrá-las.

Pela tabela 4.5, podemos observar que os algoritmos ILS encontram uma solução, em média, a menos de 1% da melhor solução encontrada, consumindo um tempo de CPU razoavelmente pequeno.

Tabela 4.5: Percentual médio da diferença entre a melhor e a segunda melhor solução encontradas pelos algoritmos e o tempo de CPU médio gastos para encontrar a segunda melhor solução para as instâncias de pequeno porte.

Algoritmos		8 pontos em $\Re^3$	9 pontos em $\Re^3$	10 pontos em $\Re^3$	11 pontos em $\Re^3$	10 pontos em $\Re^4$	10 pontos em $\Re^5$
ILS1	Percentual de desvio	0.09838	0.13603	0.12643	0.13330	0.15686	0.16923
	Desvio padrão do percentual	0.38529	0.42231	0.35860	0.39244	0.38294	0.39476
	Tempo Médio Segunda Melhor Solução	0.00087	0.00196	0.00371	0.00646	0.00591	0.00657
ILS2	Percentual de desvio	0.04287	0.05890	0.08200	0.07296	0.06780	0.05570
	Desvio padrão do percentual	0.24968	0.27797	0.32020	0.31143	0.25552	0.20009
	Tempo Médio Segunda Melhor Solução	0.00130	0.00342	0.00750	0.01160	0.00920	0.00887
ILS3	Percentual de desvio	0.09966	0.14327	0.13072	0.14900	0.15924	0.16206
	Desvio padrão do percentual	0.37965	0.41738	0.37961	0.42406	0.38656	0.35577
	Tempo Médio Segunda Melhor Solução	0.00077	0.00227	0.00377	0.00640	0.00545	0.00607
ILS4	Percentual de desvio	0.19477	0.36651	0.44320	0.57862	0.32264	0.31899
	Desvio padrão do percentual	0.81957	1.15158	1.33931	1.43763	0.85898	0.87436
	Tempo Médio Segunda Melhor Solução	0.00219	0.00626	0.01216	0.02320	0.01513	0.01694
ILS5	Percentual de desvio	0.23436	0.32986	0.42387	0.48559	0.29985	0.25425
	Desvio padrão do percentual	0.79722	1.10463	1.18549	1.22655	0.75566	0.58585
	Tempo Médio Segunda Melhor Solução	0.00224	0.00586	0.01183	0.02202	0.01536	0.01670
ILS6	Percentual de desvio	0.19421	0.36110	0.43703	0.51126	0.32545	0.27396
	Desvio padrão do percentual	0.81948	1.12192	1.33621	1.45673	0.85955	0.66847
	Tempo Médio Segunda Melhor Solução	0.00217	0.00636	0.01228	0.02196	0.01505	0.01666
micro O	Percentual de desvio	0.11142	0.13334	0.18986	0.21925	0.20755	0.19385
	Desvio padrão do percentual	0.44371	0.43191	0.51652	0.48613	0.47872	0.43954
	Tempo Médio Segunda Melhor Solução	0.00322	0.00744	0.01460	0.03021	0.02161	0.02372

A figura 4.3 apresenta o total de soluções ótimas encontradas, considerando todas iterações de cada um dos algoritmos.

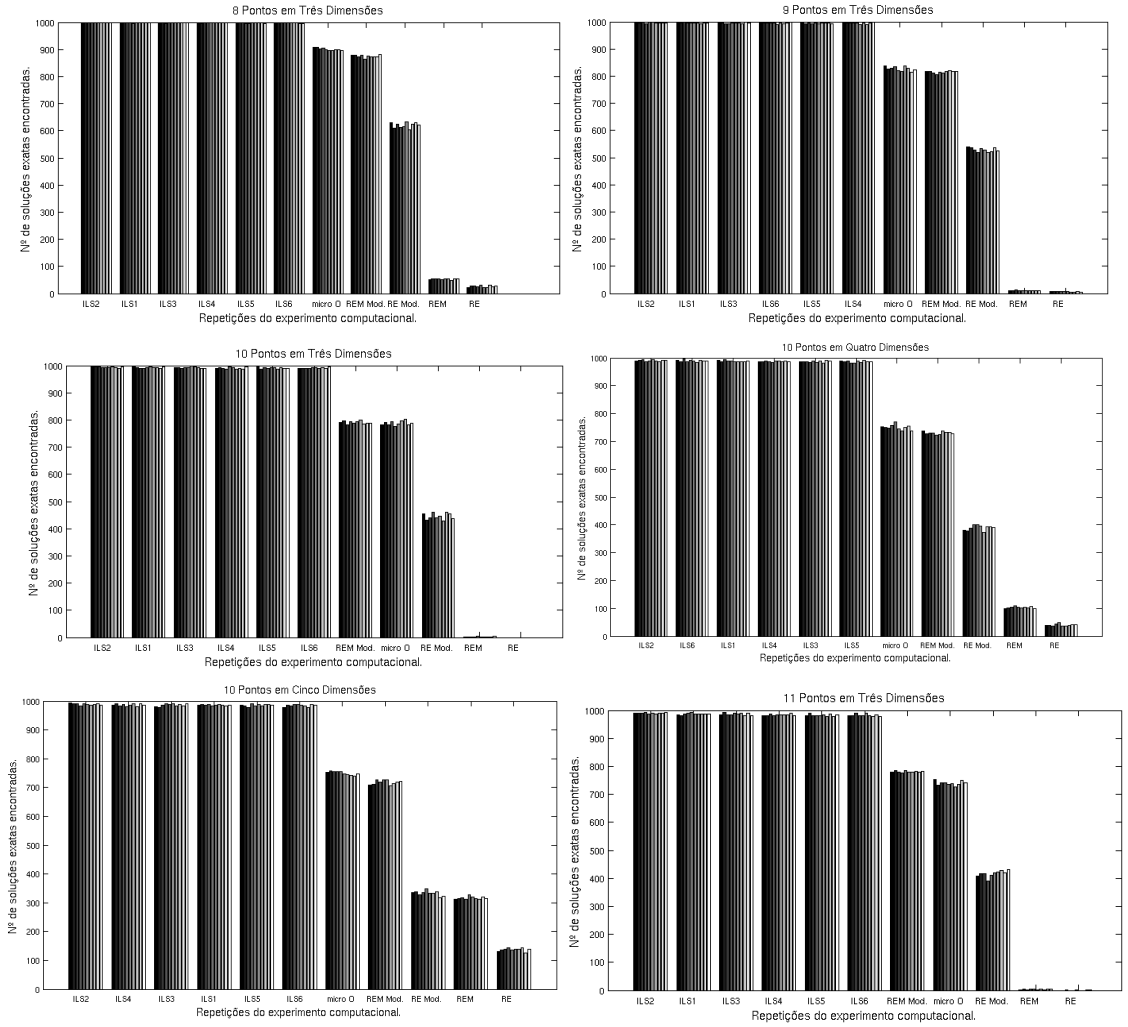

Figura 4.3: Gráficos contendo o número de soluções exatas encontradas por cada algoritmo em cada uma das dez repetições do experimento para instâncias de pequeno porte.

Analisando em conjunto todos os dados apresentados para instâncias de pequeno porte, podemos ver que, comparados com os algoritmos presentes na literatura, os algoritmos ILS obtiveram resultados melhores para instâncias de pequeno porte. Dentre os algoritmos ILS, o algoritmo ILS2 obteve resultados melhores, ou seja, mais próximos dos obtidos pelo algoritmo exato.

Para os algoritmos baseados em Relaxação Dinâmica Estendida, os resultados indicam que o algoritmo REM Mod. apresenta um melhora significativa na qualidade das soluções encontradas, sem a utilização de um tempo de CPU excessivo para obter

tal melhora.

4.4.2 Instâncias de Médio Porte

Apresentamos a seguir as tabelas e análises referentes aos resultados produzidos pelos algoritmos ILS, GRASP, GRASP com Path-relinking (GRASP-PR), Microcanônico (Micro O) e de Relaxação para as instâncias de médio porte.

Nas tabelas, os valores que não puderam ser obtidos foram substituídos por traços.

Na tabela 4.6, são reportados os desvios padrão e as médias referentes aos valores de ρ produzidos por cada algoritmo quando aplicados a cada conjunto de instâncias de médio porte.

Como podemos observar neste tabela, para as instâncias contendo 50 pontos dados em $\Re^3$, ambos os algoritmos GRASP e GRASP-PR apresentaram a média do valor de ρ menor do que as médias encontradas pelos demais algoritmos. Para as instâncias contendo 100 pontos em $\Re^3$, o algoritmo GRASP-PR, dentre todos os algoritmos, foi o que apresentou a menor média. Já para as instâncias contendo 250 pontos em $\Re^3$, o menor valor de média está associado ao algoritmo de Relaxação Dinâmica Estendida Modificada com Mult-Start (REM Mod.).

Ao compararmos separadamente as médias referentes aos algoritmos ILS, verificamos que para as instâncias com 50 pontos e 250 pontos o algoritmo ILS2 foi o que apresentou a menor média do valor de ρ. No entanto, para as instâncias com 100 pontos, o algoritmo ILS1 obteve a menor média. Verificamos que, à medida que a quantidade de pontos dados em uma instância aumenta, a média dos valores de ρ produzidos pelos algoritmos ILS que utilizam o procedimento guloso passam a ser maiores que 1. Dessa forma, as árvores produzidas por estes algoritmos possuem tamanho maior que o tamanho da árvore geradora mínima, e consequentemente, são consideradas soluções ruins para o PSE.

Tabela 4.6: Média e desvio padrão dos valores de ρ obtidos por cada algoritmo aplicado nas instâncias de médio porte.

Algoritmos	Medidas	50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
ILS1	Média	0.94777	0.94932	0.95125
	Desvio padrão	0.00916	0.00278	0.00410
ILS2	Média	0.94756	0.94965	0.95094
	Desvio padrão	0.00905	0.00404	0.00441
ILS3	Média	0.94825	0.95165	0.95687
	Desvio padrão	0.00936	0.00289	0.00384
ILS4	Média	0.94828	0.95744	1.02283
	Desvio padrão	0.00862	0.00663	0.02323
ILS5	Média	0.94814	0.95806	1.02496
	Desvio padrão	0.00863	0.00569	0.01228
ILS6	Média	0.94962	0.96798	1.03573
	Desvio padrão	0.00811	0.00898	0.01241
GRASP	Média	0.94079	0.94896	0.96547
	Desvio padrão	0.02164	0.02203	0.02966
GRASP-PR	Média	0.94079	0.939641	0.95933
	Desvio padrão	0.02164	0.02188	0.02446
micro O	Média	0.95121	0.95410	0.95427
	Desvio padrão	0.00801	0.00438	0.00354
RE	Média	0.95279	0.95462	0.95290
	Desvio padrão	0.00836	0.00502	0.00373
REM	Média	0.95024	0.95229	0.95095
	Desvio padrão	0.00831	0.00458	0.00356
RE Mod.	Média	0.95276	0.95458	0.95285
	Desvio padrão	0.00836	0.00501	0.00373
REM Mod.	Média	0.95021	0.95226	0.95091
	Desvio padrão	0.00831	0.00458	0.00356

Na tabela 4.7 são reportados as medidas resumo associadas aos valores de ρ, obtidos através da aplicação dos algoritmos para instâncias de médio porte.

Tabela 4.7: Resumo dos valores de ρ obtidos por cada algoritmo aplicado a instâncias de médio porte.

Algoritmos		50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
ILS1	Máximo	0.95943	0.95417	0.95798
	3º quartil	0.95490	0.95210	0.95379
	Mediana	0.94739	0.94929	0.95097
	1º quartil	0.94411	0.94722	0.94899
	Mínimo	0.92709	0.94435	0.94288
ILS2	Máximo	0.95943	0.95973	0.95934
	3º quartil	0.95490	0.95150	0.95433
	Mediana	0.94815	0.94929	0.94987
	1º quartil	0.94411	0.94730	0.94761
	Mínimo	0.92709	0.94307	0.94301
ILS3	Máximo	0.96054	0.95825	0.96384
	3º quartil	0.95578	0.95352	0.95882
	Mediana	0.94886	0.95121	0.95670
	1º quartil	0.94411	0.95028	0.95424
	Mínimo	0.92725	0.94655	0.95039
ILS4	Máximo	0.95943	0.97310	1.07189
	3º quartil	0.95547	0.96041	1.03757
	Mediana	0.95040	0.95666	1.02096
	1º quartil	0.94411	0.95273	1.00453
	Mínimo	0.92969	0.94627	0.98520
ILS5	Máximo	0.96197	0.97017	1.04272
	3º quartil	0.95566	0.96251	1.03557
	Mediana	0.94739	0.95785	1.03004
	1º quartil	0.94411	0.95305	1.01488
	Mínimo	0.93124	0.94979	1.00012
ILS6	Máximo	0.95943	0.98915	1.05400
	3º quartil	0.95642	0.97106	1.04612
	Mediana	0.94786	0.96683	1.03377
	1º quartil	0.94528	0.96420	1.03020
	Mínimo	0.93415	0.95347	1.01632
micro O	Máximo	0.96367	0.96497	0.96015
	3º quartil	0.95606	0.95576	0.95797
	Mediana	0.95498	0.95341	0.95374
	1º quartil	0.94507	0.95080	0.95250

Continua na próxima página.

Tabela 4.7 – continuação da página anterior.

Algoritmos		50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
	Mínimo	0.93222	0.94834	0.94791
RE	Máximo	0.96154	0.96344	0.96088
	3º quartil	0.95929	0.95803	0.95343
	Mediana	0.95605	0.95492	0.95320
	1º quartil	0.94938	0.95169	0.95120
	Mínimo	0.92938	0.94624	0.94377
REM	Máximo	0.96057	0.96108	0.95827
	3º quartil	0.95544	0.95628	0.95235
	Mediana	0.95256	0.95125	0.95092
	1º quartil	0.94596	0.94934	0.94959
	Mínimo	0.92809	0.94517	0.94245
RE Mod.	Máximo	0.96151	0.96341	0.96083
	3º quartil	0.95927	0.95798	0.95340
	Mediana	0.95602	0.95488	0.95317
	1º quartil	0.94935	0.95166	0.95117
	Mínimo	0.92935	0.94621	0.94373
REM Mod.	Máximo	0.96054	0.96104	0.95823
	3º quartil	0.95542	0.95624	0.95231
	Mediana	0.95253	0.95122	0.95088
	1º quartil	0.94594	0.94930	0.94956
	Mínimo	0.92807	0.94513	0.94241

A tabela 4.8 traz o tempo médio total de CPU consumido por cada um dos algoritmos até que o seu critério de parada fosse satisfeito e o tempo de CPU consumido para encontrar a melhor solução.

Com base nesta tabela, temos que um elevado tempo de CPU é consumido pelos algoritmos ILS quando aplicados a instâncias de médio porte. Este fato deve-se à alta complexidade do problema, que faz com que o número de topologias possíveis cresça de forma super-exponencial à medida que a quantidade de pontos dados aumenta, além do aumento do esforço computacional para obter o posicionamento ótimo dos pontos de Steiner em uma árvore com uma quantidade consideravelmente grande de pontos.

Ao compararmos todos os algoritmos que tem como base a busca local em ve-

tores topologia, percebemos que o menor tempo de CPU é atribuído ao algoritmo GRASP. Entretanto, não podemos justificar o motivo pelo qual tal algoritmo encontra soluções em um tempo de CPU consideravelmente menor, uma vez que não há detalhes sobre como o tempo computacional foi aferido.

Com relação a todos os algoritmos, os algoritmos de Relaxação foram os que produziram soluções com menor tempo de CPU.

Tabela 4.8: Tempo médio de CPU em segundos gastos no total e para encontrar a melhor solução reportada para instâncias de médio porte.

Algoritmos		50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
ILS1	Tempo Médio Total	153.43679	1471.10287	21184.93198
	Tempo Médio Melhor Solução	53.96524	865.99252	17128.79368
ILS2	Tempo Médio Total	165.80770	1566.32722	24125.79897
	Tempo Médio Melhor Solução	69.78608	992.67030	19979.01794
ILS3	Tempo Médio Total	116.00645	668.46924	3186.06952
	Tempo Médio Melhor Solução	12.39597	30.16322	171.78702
ILS4	Tempo Médio Total	384.36215	3393.39181	31649.35076
	Tempo Médio Melhor Solução	228.77163	2243.82796	21381.73841
ILS5	Tempo Médio Total	368.63317	3479.20650	25061.61185
	Tempo Médio Melhor Solução	207.52657	2334.44536	15989.87344
ILS6	Tempo Médio Total	309.05371	2272.99699	24226.01456
	Tempo Médio Melhor Solução	146.32061	1084.42751	13393.12822
micro O	Tempo Médio Total	89.59120	328.75415	2786.07679
	Tempo Médio Melhor Solução	54.20739	217.51919	2222.20261
GRASP	Tempo Médio Total	6.67937	12.96968	37.68156
	Tempo Médio Melhor Solução	-	-	-
GRASP-PR	Tempo Médio Total	7.37281	14.32312	40.76812
	Tempo Médio Melhor Solução	-	-	-
RE	Tempo Médio Total	0.00427	0.00720	0.12027
	Tempo Médio Melhor Solução	-	-	-
REM	Tempo Médio Total	0.32187	0.65493	1.86827
	Tempo Médio Melhor Solução	-	-	-
RE Mod.	Tempo Médio Total	0.00613	0.00907	0.02987
	Tempo Médio Melhor Solução	-	-	-
REM Mod.	Tempo Médio Total	0.32107	0.68400	1.74080

Continua na próxima página.

Tabela 4.8 – continuação da página anterior.

Algoritmos		50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
	Tempo Médio Melhor Solução	-	-	-

Na tabela 4.9 são apresentados o desvio padrão e a média do percentual da diferença entre os valores de ρ para a melhor solução encontrada e a segunda melhor solução. Também é apresentado o tempo médio de CPU gasto para encontrar a segunda melhor solução.

Nesta tabela podemos observar que é possível encontrar soluções bem próximas, a menos de 1%, em média, das melhores soluções produzidas em um tempo de CPU razoável.

Tabela 4.9: Percentual médio da diferença entre a melhor e a segunda melhor solução encontradas pelos algoritmos e o tempo de CPU médio gastos para encontrar a segunda melhor solução para as instâncias de médio porte.

Algoritmos		50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
ILS1	Percentual de desvio	0.03102	0.05762	0.06647
	Desvio padrão do percentual	0.04457	0.09166	0.15782
	Tempo Médio Segunda Melhor Solução	33.42182	680.83375	15647.32110
ILS2	Percentual de desvio	0.05934	0.05594	0.15705
	Desvio padrão do percentual	0.08343	0.08673	0.25523
	Tempo Médio Segunda Melhor Solução	32.46917	733.58611	14418.23922
ILS3	Percentual de desvio	0.06158	0.01461	0.02593
	Desvio padrão do percentual	0.11001	0.04411	0.05909
	Tempo Médio Segunda Melhor Solução	6.53561	9.85075	106.06006
ILS4	Percentual de desvio	1.05428	0.97891	2.59382
	Desvio padrão do percentual	2.61303	2.50521	4.74543
	Tempo Médio Segunda Melhor Solução	125.35370	1732.31093	17215.75805
ILS5	Percentual de desvio	0.50231	0.85120	3.22014
	Desvio padrão do percentual	1.52207	2.65113	5.04231
	Tempo Médio Segunda Melhor Solução	148.85917	1810.58675	8937.06546
ILS6	Percentual de desvio	0.37003	1.14052	2.55119
	Desvio padrão do percentual	0.71334	1.84737	3.49087

Continua na próxima página.

Tabela 4.9 – continuação da página anterior.

Algoritmos		50 pontos em $\Re^3$	100 pontos em $\Re^3$	250 pontos em $\Re^3$
	Tempo Médio Segunda Melhor Solução	101.43194	824.79448	7946.48036
micro O	Percentual de desvio	0.38564	0.27087	0.34215
	Desvio padrão do percentual	0.65677	0.56974	0.65781
	Tempo Médio Segunda Melhor Solução	38.27973	178.33514	1690.17016

Analisando em conjunto todos os dados apresentados para instâncias de médio porte, podemos verificar que os algoritmos ILS produziram soluções de boa qualidade. Entretanto, necessitaram de elevado tempo computacional para encontrá-las. Dentre os algoritmos ILS, o algoritmo ILS2, em geral, obteve soluções de melhor qualidade, ou seja, menor média dos valores de ρ.

Notamos, também, que os algoritmos baseados em Relaxação Dinâmica Estendida produziram soluções de boa qualidade, com a utilização de baixo tempo computacional. A obtenção da menor média dos valores de ρ para instâncias com 250 pontos indica que o algoritmo $REMMod.$ representa uma alternativa viável para a obtenção de soluções de boa qualidade para instâncias de grande porte.

4.4.3 Instâncias de Grande Porte

Apresentamos a seguir os resultados, tabelas e análises referentes aos resultados produzidos pelos algoritmos de Relaxação Dinâmica Estendida para instâncias de grande porte.

Na tabela 4.10, são apresentados os desvios e as médias associados aos valores de ρ produzidos pelos algoritmos ao serem aplicados a instâncias de grande porte.

Nessa tabela, podemos verificar que os algoritmos de Relaxação Dinâmica Estendida produziram soluções de boa qualidade para o PSE, sendo a versão com modificação proposta e com o Mult-Start (REM Mod.) a que obteve, em média, o melhor resultado.

Tabela 4.10: Média e desvio padrão dos valores de ρ obtidos por cada algoritmo aplicado nas instâncias de grande porte.

Algoritmos	Medidas	1000 pontos em $\Re^3$	10000 pontos em $\Re^3$	100000 pontos em $\Re^3$
RE	Média	0.95412	0.95420	0.95426
	Desvio padrão	0.00150	0.00050	0.00015
REM	Média	0.95307	0.95379	0.95415
	Desvio padrão	0.00144	0.00050	0.00015
RE Mod.	Média	0.95408	0.95417	0.95423
	Desvio padrão	0.00150	0.00050	0.00015
REM Mod.	Média	0.95303	0.95376	0.95411
	Desvio padrão	0.00144	0.00050	0.00015

Na tabela 4.11, apresentamos as medidas resumos associadas aos valores de ρ, obtidos pelos algoritmos para instâncias de grande porte.

Tabela 4.11: Resumo dos valores de ρ obtidos por cada algoritmo aplicado a instâncias de grande porte.

		1000 pontos em $\Re^3$	10000 pontos em $\Re^3$	100000 pontos em $\Re^3$
RE	Máximo	0.95742	0.95553	0.95461
	3º quartil	0.95523	0.95457	0.95438
	Mediana	0.95390	0.95416	0.95425
	1º quartil	0.95312	0.95379	0.95416
	Mínimo	0.94992	0.95326	0.95394
REM	Máximo	0.95634	0.95500	0.95451
	3º quartil	0.95423	0.95416	0.95426
	Mediana	0.95299	0.95373	0.95415
	1º quartil	0.95213	0.95342	0.95404
	Mínimo	0.94929	0.95286	0.95383
RE Mod.	Máximo	0.95738	0.95550	0.95457
	3º quartil	0.95520	0.95454	0.95434
	Mediana	0.95387	0.95413	0.95421
	1º quartil	0.95308	0.95376	0.95413
	Mínimo	0.94989	0.95323	0.95390

Continua na próxima página.

Tabela 4.11 – continuação da página anterior.

		1000 pontos em $\Re^3$	10000 pontos em $\Re^3$	100000 pontos em $\Re^3$
REM Mod.	Máximo	0.95631	0.95497	0.95447
	3º quartil	0.95420	0.95413	0.95422
	Mediana	0.95295	0.95370	0.95412
	1º quartil	0.95209	0.95338	0.95400
	Mínimo	0.94925	0.95283	0.95380

Na tabela 4.12, apresentamos o tempo médio de CPU consumido pelo procedimento de construção da AGM e o tempo médio total de CPU consumido por cada algoritmo até que o seu critério de parada seja atingido.

Podemos observar, nessa tabela, que a versão Mult-Start dos algoritmos consumiu um tempo de CPU expressivamente maior que o consumido pelos demais algoritmos. No entanto, dada a complexidade do problema, os tempos de CPU são considerados satisfatorios. Também verificamos que os algoritmos que foram implementados com a modificação proposta apresentaram um tempo computacional compatível com os outros algoritmos.

Tabela 4.12: Tempo médio de CPU em segundos gastos no total e para encontrar a melhor solução reportada para instâncias de grande porte.

		1000 pontos em $\Re^3$	10000 pontos em $\Re^3$	100000 pontos em $\Re^3$
	Tempo Médio AGM	0.02136	1.03204	105.77708
RE	Tempo Médio Total	0.21704	3.07892	236.47818
REM	Tempo Médio Total	7.62874	83.15304	2772.10086
RE Mod.	Tempo Médio Total	0.11892	2.87424	236.85222
REM Mod.	Tempo Médio Total	7.66111	83.78864	2895.49914

Analisando em conjunto os dados obtidos para instâncias de grande porte, verificamos que os algoritmos de Relaxação Dinâmica Estendida produziram soluções de ótima qualidade. Dada a dimensão das instâncias, ressaltamos que as soluções

foram obtidas em um tempo de CPU bastante reduzido.

Capítulo 5

Conclusões

Neste trabalho, desenvolvemos algoritmos heurísticos para a obtenção de soluções de boa qualidade para o Problema de Steiner Euclidiano em $\Re^n$. Foram desenvolvidos algoritmos baseados na metaheurística ILS que incorporaram a busca local em vetores topologia e algoritmos referentes a uma melhoria proposta ao algoritmo de Relaxação Dinâmica Estendida. O desempenho de cada algoritmo foi avaliado através de resultados de experimentos computacionais com instâncias de pequeno, médio e grande portes. Tais instâncias foram geradas aleatoriamente em um cubo ou hiper-cubo unitário com distribuição uniforme.

A partir do estudo da metaheurística ILS foram desenvolvidos dois procedimentos para construção de uma solução inicial. Um constrói o vetor topologia inicial de forma gulosa, e o outro constrói o vetor topologia através da inserção de pontos de Steiner na árvore geradora mínima, obtida para os pontos dados. Também foram desenvolvidos três procedimentos de aceitação, quais sejam: aceitar o melhor candidato, aceitar o candidato com valor pertencente a um intervalo construído a partir de soluções pertencentes a um conjunto elite e aceitar o candidato probabilisticamente. A combinação desses procedimentos produziu 6 algoritmos ILS.

Realizamos também experimentos com o objetivo de melhor ajustar os parâmentros contidos nos algoritmos heurísticos implementados. Para a maioria dos parâmetros, os experimentos consistirão em exaustivas execuções dos algoritmos para diversos valores dos parâmetros. Para o parâmetro do tamanho da perturbação associado ao procedimento de perturbação contido nos algoritmos ILS, foi desenvolvido um experimento que consiste na execução de um algoritmo simplificado

utilizado para avaliar o impacto de alguns valores para esse parâmetro na solução produzida, sendo os resultados obtidos utilizados para inferir seu valor.

Após analisar os resultados obtidos pelo experimento computacional, verificamos que os algoritmos ILS implementados obtiveram êxito para instâncias de pequeno porte (com até 11 pontos), obtendo uma quantidade maior de soluções ótimas quando comparadas com os demais algoritmos propostos na literatura e demandando um pequeno tempo computacional, da ordem de 0.03 segundo, para encontrá-las.

Já para instâncias de médio porte (com até 250 pontos), foi possível observar que os algoritmos ILS que utilizaram o procedimento guloso para construção de solução inicial não conseguiram encontrar soluções de boa qualidade, produzindo soluções com valor de ρ maior que 1, ou seja, árvores com tamanho maior que o da AGM contruída para os pontos dados. Os demais algoritmos ILS encontraram soluções de boa qualidade. Quando as soluções encontradas em média não são as menores, as soluções estão, em média, a no máximo 1% da melhor solução presente na literatura. Essas soluções foram encontradas ao custo de um grande tempo computacional, onde o menor tempo CPU foi da ordem de 116 segundos e o maior tempo de CPU da ordem de 24125 segundos. No entanto, Soluções, em média, a menos 0.1% das soluç oes reportadas podem ser encontradas com no máximo 14418 segundos de tempo de CPU.

Dentre os algoritmos ILS implementados, o algoritmo que utilizou os procedimentos de inserção de pontos de Steiner na AGM para a obtenção de uma solução inicial e de aceitação utilizando o conjunto elite foi o que obteve melhor performance, tanto para instâncias de pequeno porte quanto para instâncias de médio porte.

Com relação aos algoritmos baseados em Relaxação Dinâmica Estendida, verificamos que os algoritmos com a modificação proposta nesta dissertação encontraram um número expressivo de soluções ótimas, de 727 à 881 soluções ótimas de 1000 instâncias, para instâncias de pequeno porte. Para instâncias de médio porte, quando comparadas aos outros algoritmos, estas obtiveram soluções bem próximas das melhores.

Em particular, para instâncias com até 250 pontos, a versão Mult-Start com a modificação produziu soluções de melhor qualidade quando comparadas às produzidas pelos demais algoritmos, consumindo um tempo de CPU bem menor, de aproxi-

madamente 1.5 segundo. Para instâncias de grande porte, foram obtidas soluções de boa qualidade, sendo o valor ρ encontrado por todas as instâncias menor que 1. Para todas as instâncias de grande porte, os valores de ρ encontrados foram menores que 0.96. Essas soluções foram produzidas em um tempo que, em média, variou de 0.12 segundo (para instâncias com 1000 pontos) a 2895 segundos (para instâncias com 100.000 pontos). Dada a alta complexidade do problema e a quantidade de pontos dados nas instâncias, podemos afirmar que o tempo computacional requerido por esses algoritmos foi pequeno.

Observa-se, ainda, que para todas as instâncias de pequeno, médio e grande portes, o algoritmo baseado na heurística de Relaxação Dinâmica Estendida (versão Mult-Start) com a modificação proposta neste trabalho foi o que, para todas as instâncias, encontrou a árvore de Steiner de menor tamanho.

Os resultados produzidos nesta dissertação indicam a possibilidade de, em trabalhos futuros, desenvolver algoritmos heurísticos para obter soluções de melhor qualidade para instâncias ainda maiores do PSE em $\Re^n$.

A seguir, listamos alguns dos trabalhos que poderão ser realizados futuramente:

- Aplicar os algoritmos Mult-Start de Relaxação Dinâmica Estendida a instâncias com milhares de pontos.
- Utilizar outras técnicas para o ajuste dos parâmetros.
- Modificar os procedimentos de busca local e perturbação contidos nos algoritmos ILS propostos, inserindo técnicas reativas visando encontrar solução de qualidade melhor em menor tempo de CPU quando aplicados a instâncias de médio porte.
- Desenvolver algoritmos que utilizem outras metaheurísticas, como VNS(variable neighborhood search), em conjunto com a versão da busca local em vetores topologia desenvolvida neste trabalho.
- Estudo de metodologias diferentes para o posicionamento ótimo de pontos de Steiner em uma árvore com topologia de Steiner.
- Desenvolver aplicações dos algoritmos em novos problemas de agrupamento (*cluster analysis*) e outras aplicações.

Referências Bibliográficas

[1] SMITH, W. D., SMITH, J. M., "On the Steiner Ratio in 3-Space", *J. Combinatorial Theory*, v. 69, pp. 301–332, 1995.

[2] GAREY, M. R., GRAHAM, R. L., JOHNSON, D. S., "The Complexity of Computing Steiner Minimal Trees", *SIAM J. Appl. Math.*, v. 32, pp. 835–859, 1977.

[3] KUHN, H. W., "Steiners's Problem Revisited", In: DANTZIG, EAVES, B. C. (eds), *Studies in Optimization, Studies in Mathematics*, v. 10, pp. 53–70, Math. Assoc. Amer., 1975.

[4] CIESLIK, D., *Steiner Minimal Trees*. v. 23. *Nonconvex Optimization and Its Application*, Kluwer acadeemic Publishers, 1998.

[5] MACULAN, N., "The Steiner Tree Problem in Graphs", *Annals of discrete Mathematics*, v. 31, pp. 185–212, 1987.

[6] HANAN, M., "On Steiner's Problem with Rectilinear Distance", *SIAM Journal on Applied mathematics*, v. 14, pp. 255–265, 1966.

[7] GILBERT, E. N., POLLAK, H. O., "Steiner Minimal Trees", *SIAM Journal on Applied mathematics*, v. 16, pp. 323–345, 1968.

[8] HWANG, F. K., RICHARDS, D. S., WINTER, P., "The Steiner Tree Problem", In: *Annals of Discrete Mathematics*, v. 53, North-Holland, Amsterdan, 1992.

[9] SMITH, W. D., "How to Find Steiner Minimal Tree in Euclidean d-Space", *Algorithmica*, v. 7, n. 2/3, pp. 137–178, 1992.

[10] MONTENEGRO, F. M. T., *Heurísticas Para o Problema de Steiner Euclidiano em $\Re^n$*, Ph.D. Thesis, COPPE - Universidade Federal do Rio de Janeiro, 2001.

[11] GAREY, M. R., JOHNSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.

[12] MACULAN, N., MICHELON, P., XAVIER, A. E., "The Euclidean Steiner Tree Problem in $\Re^n$: a Mathematical Programming Formulation", *Annals of Operations Research*, v. 96, pp. 209–209, 2000.

[13] FAMPA, M., MACULAN, N., "Using a Conic Formulation for finding Steiner Minimal Trees." *Numerical Algorithms*, v. 35, pp. 315–330, 2004.

[14] COURANT, R., ROBBINS, H., *What is Mathematics?*. Oxford University Press, 1941.

[15] MIEHLE, W., "Link-length minimization in networks", *Operations Research*, v. 6, pp. 232–243, 1958.

[16] CHAPEU-BLONDEAU, F., JANEZ, F., FERRIER, J.-L., "A Dynamic Adapitive Relaxation Scheme Applied to the Euclidean Steiner Minimal Tree Problem", *Siam J. Optim.*, v. 7, pp. 1037–1053, 1997.

[17] SOUKUP, J., "Minimum Steiner Trees, Roots of a Polinomial, and Other Magic", *ACM/SIGMAP Newsletter*, v. 22, pp. 37–51, 1977.

[18] ROCHA, M. L., *Aplicações de Algoritmos Paralelos e Híbridos para o Problema de Árvore de Steiner Euclidiana no $\Re^n$*, Ph.D. Thesis, COPPE - Universidade Federal do Rio de Janeiro, 2008.

[19] ALFORD, C., BRAZIL, M., LEE, D. H., "Handbook of Operations Research in Natural Resources", chap. Optimization in Underground Mining., pp. 561–578, Springer, 2006.

[20] SMITH, J. M., "Handbook of Combinatorial Optimization", v. 2, chap. Steiner Minimal Tree in E^3: Theory, Algorithms and Applications., pp. 397–470, Kluwer Academic Publishers, 1998.

[21] STANTON, C., SMITH, J. M., "Steiner Trees and 3-D Macromolecular Conformation." *INFORMS J. on Computing*, v. 16, n. 4, pp. 470–485, 2004.

[22] THOMPSON, E. A., "The Method of Minimum Evolution", *Annals of Human Genetics*, v. 36, pp. 333–340, 1973.

[23] DU, D. Z., HWANG, F. K., "a Proof of the Gilbert-Pollak Conjecture on the Steiner Ratio", *Algorithmica*, v. 7, n. 2/3, pp. 121–135, 1992.

[24] DU, D. Z., SMITH, W. D., "Disproofs of Generalized Gilbert-Pollak Conjecture on the Steiner Ratio in Three or More Dimensions", *Journal of Combinatorial Theory*, v. 74, pp. 115–130, 1996.

[25] MONDAINI, R., "The Disproof of a Conjecture on the Steiner Ratio in E^3 and its consequences for a Full Geometric Description of Macromolecular Chiralit." In: *2nd Braz. Symp. Math. Comp. Biol.*, pp. 101–117, Rio de Janeiro, RJ, 2003.

[26] FAMPA, M., ANSTREICHER, K., "An Improved Algorithm for Computing Steiner Minimal Trees in Euclidean d-space." *Discrete Optimization*, v. 5, pp. 530–540, 2008.

[27] LINDEROTH, J. T., SAVELSBERGH, M. W. P., "A computational Study of Branch and Bound Search Strategies for Mixed Integer Programming", *INFORMS J. Computing*, v. 11, pp. 173–187, 1999.

[28] RAVADA, S., SHERMAN, A. T., "Experimental Evaluation of a Partitioning Algoritm for the Steiner Tree Problem in $\Re^2$ and $\Re^3$", *Networks*, v. 24, pp. 409–415, 1994.

[29] KALPAKIS, K., SHERMAN, A. T., "Probabilistic Analysis of an Enhanced Partitioning Algorithm for the Steiner Tree Problem in $\Re^d$", *Networks*, v. 24, pp. 147–159, 1994.

[30] SMITH, J. M., WEISS, R., PATEL, M., "An $O(N^2)$ Heuristic for Steiner Minimal Trees in E^3", *Networks*, v. 25, pp. 273–289, 1995.

[31] PEREIRA, M., *Proposta e Avaliação para o Problema de Steiner Geométrico em Duas e Três Dimensões*, Master's Thesis, UFPE, 1998.

[32] PREPARATA, F. P., SHAMOS, M. I., *Computational Geometry: An Introduction.*. Spring-verlag, New York, 1985.

[33] KIRKPATRICK, S., JR., C. D. G., VECCHI, M. P., "Optimization by Simulated Annealing", *Science*, v. 220, n. 4598, pp. 671–680, 1983.

[34] LUNDY, M., "Applications of the Annealing Algorithm to Combinatorial Problems in Statistics", *Biometrika*, v. 72, pp. 191–198, 1985.

[35] GLOVER, F., LAGUNA, M., *Tabu Search*. Kluwer Academic Publishers, 1997.

[36] MONTENEGRO, F., MACULAN, N., "A Genetic Algorithm for the Euclidean Steiner Tree Problem in $\Re^n$". In: *Proc. X Cong. Ibero-latinoamericano inv. Oper. CLAIO*, Cidade do México, 2000.

[37] HÜRLIMANN, T., *The Euclidean Steiner Tree Problem, Implementation of a Heuristic.*, Tech. rep., Institut Für Informatik der Universität Freiburg, 1994.

[38] MONTENEGRO, F., TORREÃO, J. R., MACULAN, N., "Microcanonical Optimization Algorithm for the Euclidean Steiner Problem in $\Re^n$ with Application to Phylogenetic Inference." *Physical Review E - Statistical Physics, Plasmas, Fluids and Related Interdisciplinary Topics*, v. 68, n. 5, pp. 056702, 2003.

[39] CREUTZ, M., "Microcanonical Monte Carlo Simulation." *Phys.Rev. Lett.*, v. 50, pp. 1411–1414, 1983.

[40] FEO, T. A., RESENDE, M. C., "Greedy Randomized Adaptive Search Procedure", *Journal of Global Optimization*, v. 6, pp. 109–133, 1995.

[41] RESENDE, M. G. C., RIBEIRO, C. C., "Metaheuristics: Progress as Real Problem Solvers." chap. GRASP with Path-Relinking: Recent Advances and Applications., pp. 29–63, Springer, 2005.

[42] MONTENEGRO, F., MACULAN, N., PLATEAU, G., et al., "New Heuristics for the Euclidean Steiner Problem in $\Re^n$", In: RIBEIRO, C., HANSEN, P. (eds), *Essays and Survays in Metaheuristics*, chap. New Heuristics for the Euclidean Steiner Problem in $\Re^n$, pp. 509–524, *Operations Research/Computer Science Interfaces series*, Kluwer Academic Publishers, Boston, 2001.

[43] LOURENÇO, H. R., MARTIN, O., STÜTZLE, T., "Iterated Local Search", In: GLOVER, F., KOCHENBERGER, G. (eds), *Handbook of Metaheuristics*, v. 57, pp. 321–353, *International Series in Operations Research & Management Science*, Kluwer Academic Publishers, Norwell, MA, 2002.

[44] MARTIN, O., OTTO, S. W., FELTEN, E. W., "Large-step Markov Chains for the Traveling Salesman Problem." *Complex Systems*, v. 5, pp. 299–326, 1991.

[45] MARTIN, O., OTTO, S. W., "Combining Simulated Annealing with Local Search Heuristics", *Annals of Operations Research*, v. 63, pp. 57–75, 1996.

[46] HANSEN, P., MLADENOVIC, N., *Variable Neighborhood Search for the p-Median*, Tech. rep., Les Cahiers du GERAD G-97-39, GERAD and École des Hautes Études Commerciales, 1997.

[47] BAUM, E. B., *Iterated Descent: A Bretter Algorithm fo Local Search in Combinatorial Optimization Problems*, manuscript, Caltech, Pasadena, CA, 1986.

[48] JOHNSON, D. S., "Local Optimization and Travelling Salesman Problem". In: *Proceedings of the 17th Colloquium on Automata, Languages, and Programming*, v. 443, pp. 446–461, Springer verlag: Berlin, 1990.

[49] BAXTER, J., "Local Optima Avoidance in Depot Location", *Journal of the Operational Research Society*, v. 32, pp. 815–819, 1981.

[50] MARTÍ, R., "Multi-Start Methods", In: GLOVER, F., KOCHENBERGER, G. (eds), *Handbook of Metaheuristics*, v. 57, pp. 355–368, *International*

Series in Operations Research & Management Science, Kluwer Academic Publishers, Norwell, MA, 2002.

[51] PRIM, R. C., "Shortest Connection Networks and Some Generalizations." *Bell System Technical Journal*, v. 36, pp. 1389–1401, 1957.

[52] HWANG, F. K., WENG, J. F., "The Shortest Network Under a Given Topology", *J. Algorithms*, v. 13, pp. 468–488, 1992.

[53] MLADENOVIC, N., HANSEN, P., "Variable Neighborhood Search", *Computers Ops. Res*, v. 24, n. 24, pp. 1097–1100, 1997.

Apêndice A

Histogramas para Ajuste do Tamanho da Perturbação

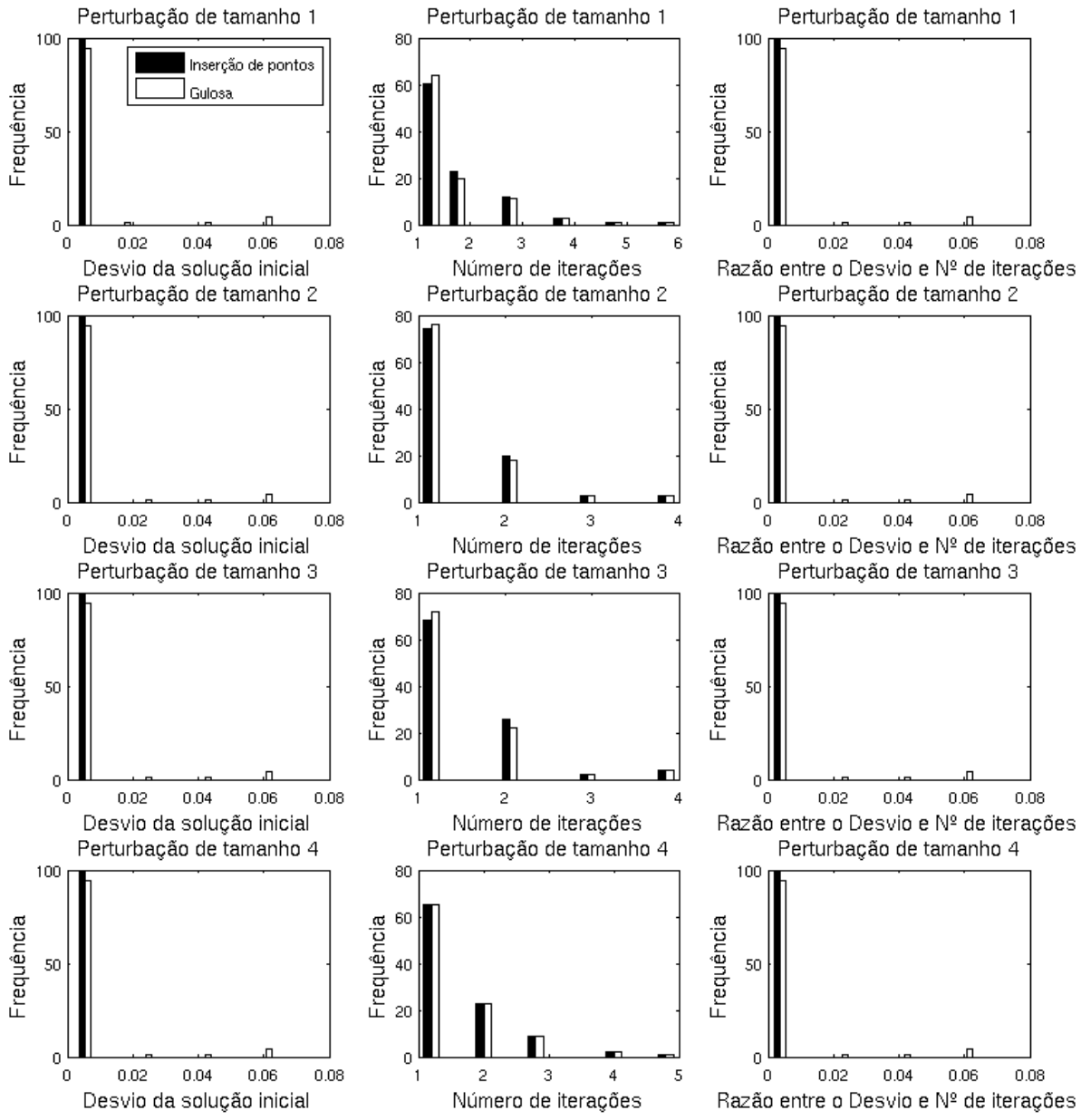

Figura A.1: Histogramas para a calibração do tamanho da perturbação para instâncias com 8 pontos dados.

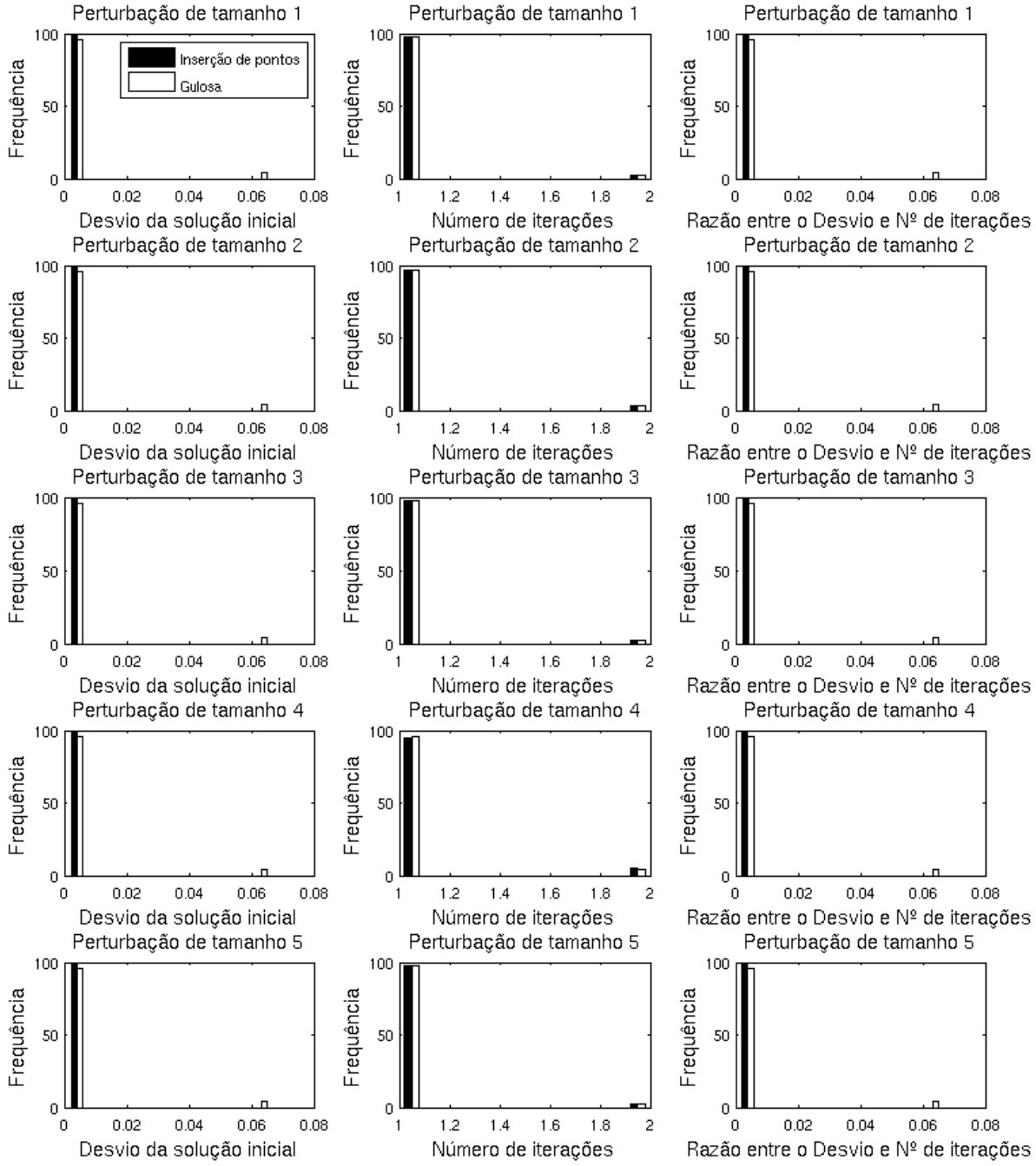

Figura A.2: Histogramas para a calibração do tamanho da perturbação para instâncias com 9 pontos dados.

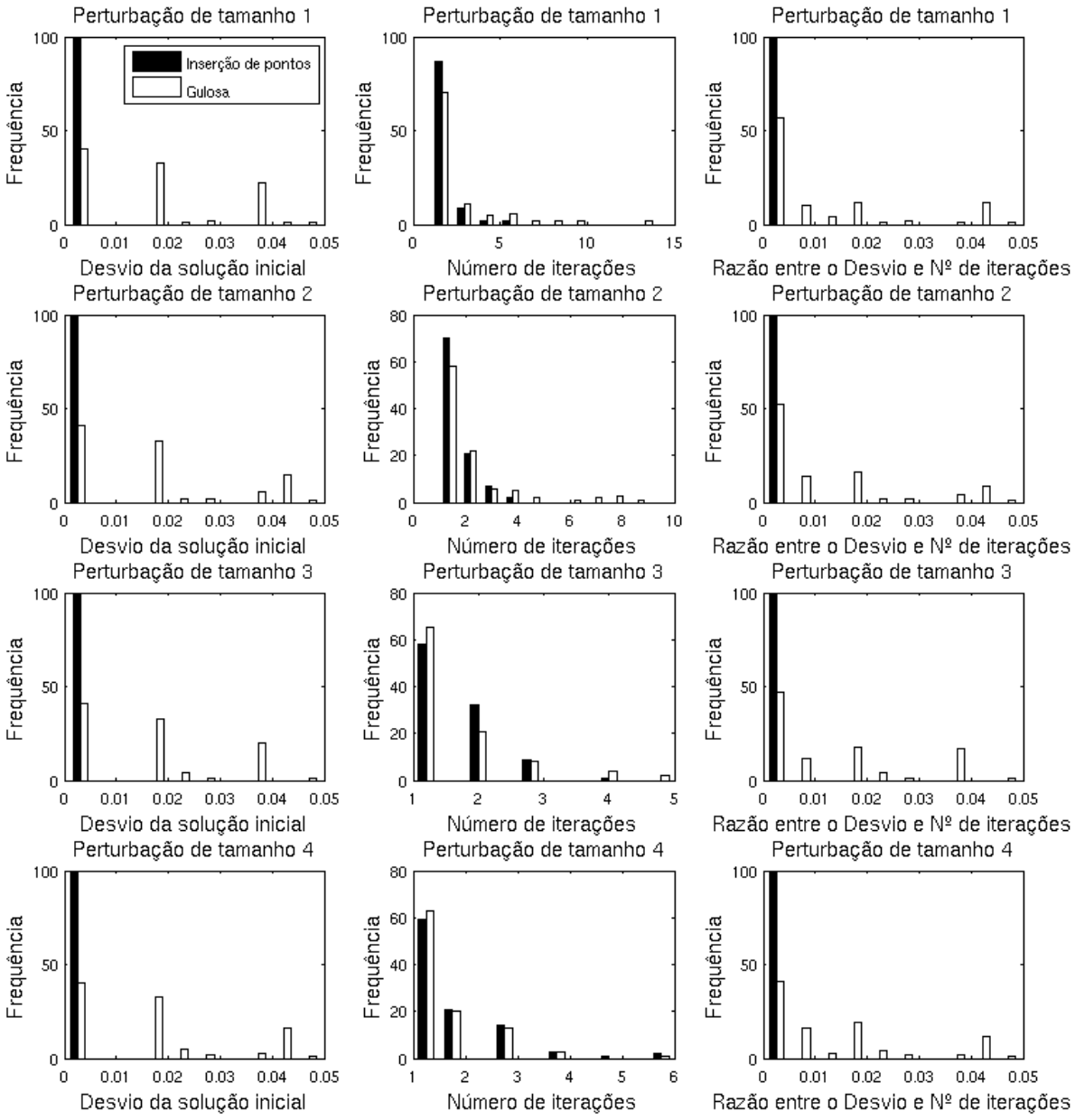

Figura A.3: Histogramas para a calibração do tamanho da perturbação para instâncias com 10 pontos dados.

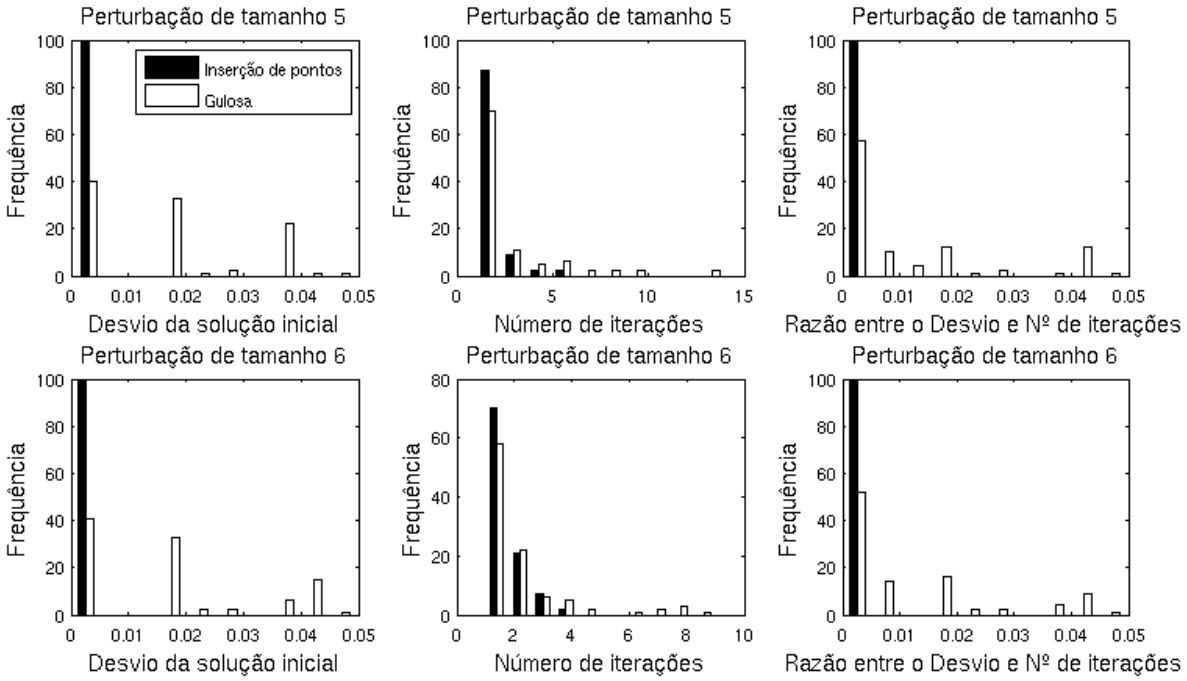

Figura A.4: Continuação da FiguraA.3 - Histogramas para a calibração do tamanho da perturbação para instâncias com 10 pontos dados.

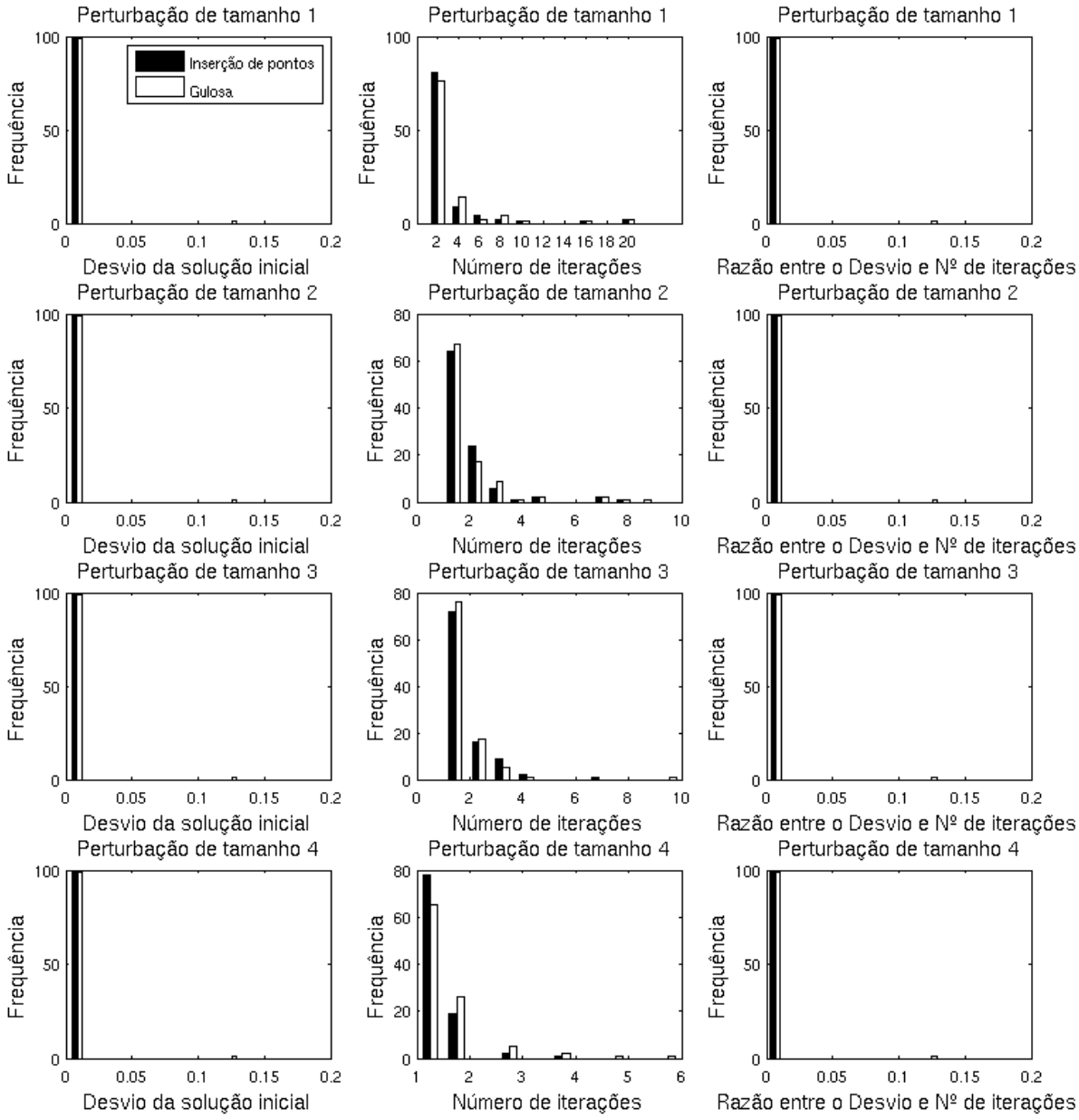

Figura A.5: Histogramas para a calibração do tamanho da perturbação para instâncias com 11 pontos dados.

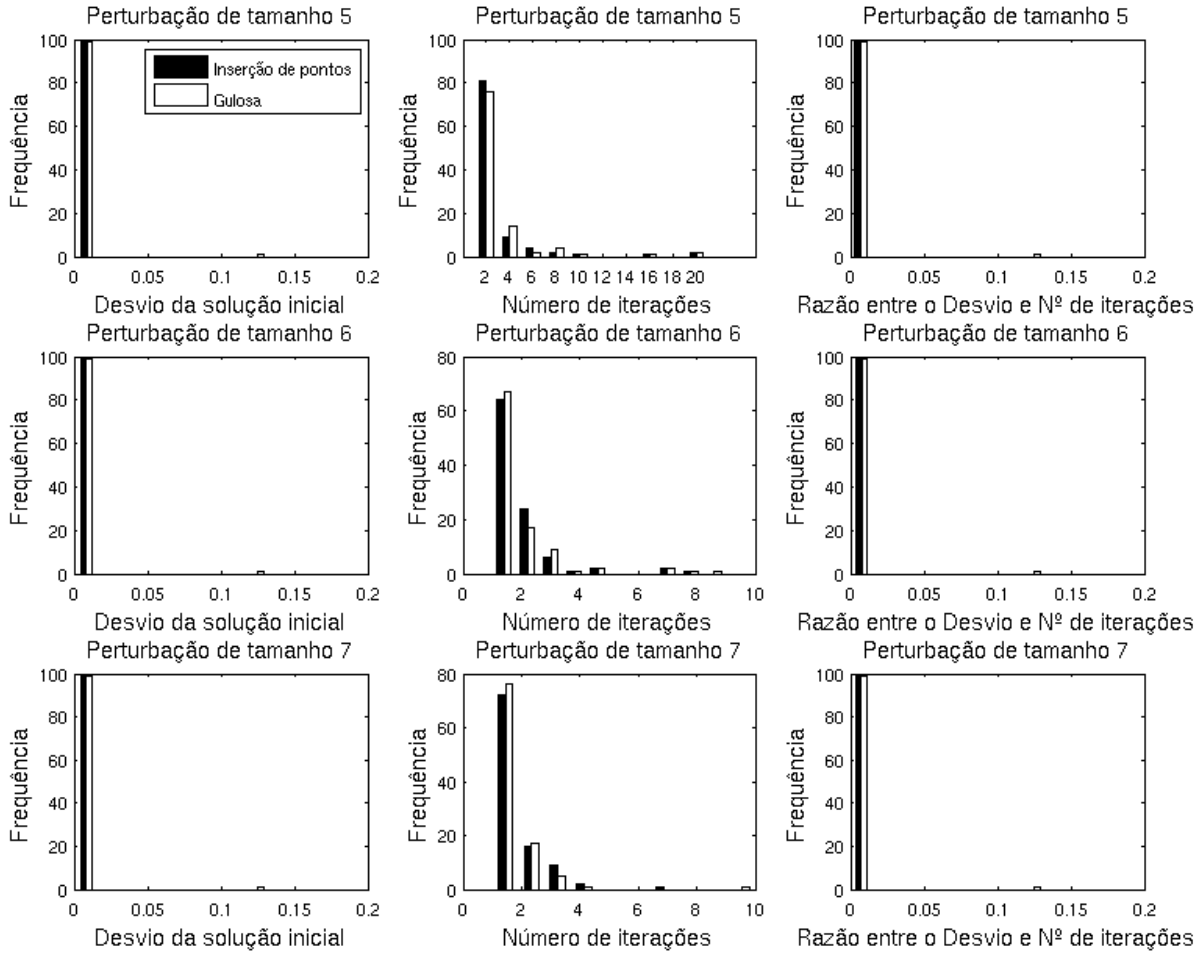

Figura A.6: Continuação da FiguraA.5 - Histogramas para a calibração do tamanho da perturbação para instâncias com 11 pontos dados.

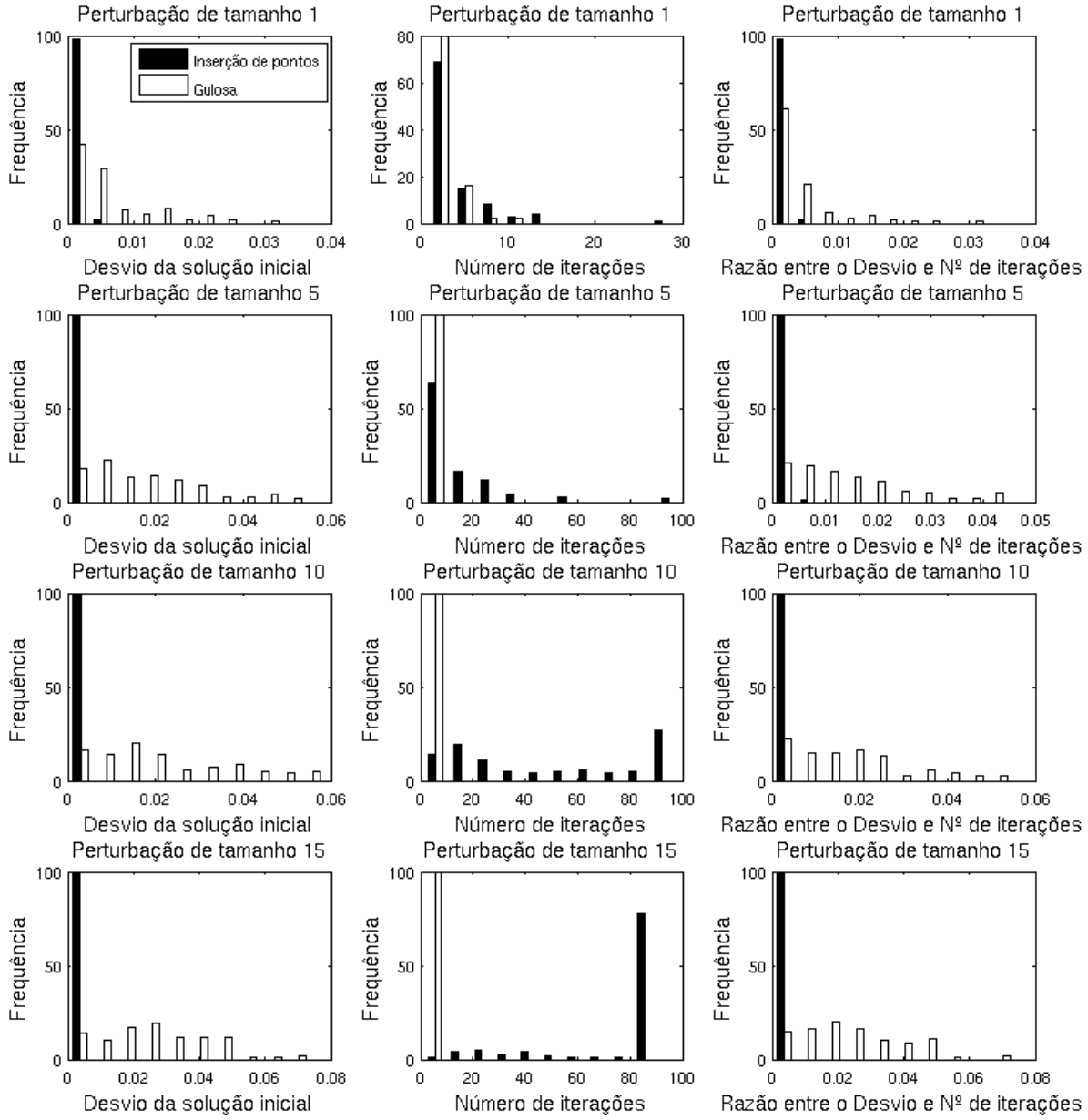

Figura A.7: Histogramas para a calibração do tamanho da perturbação para instâncias com 100 pontos dados.

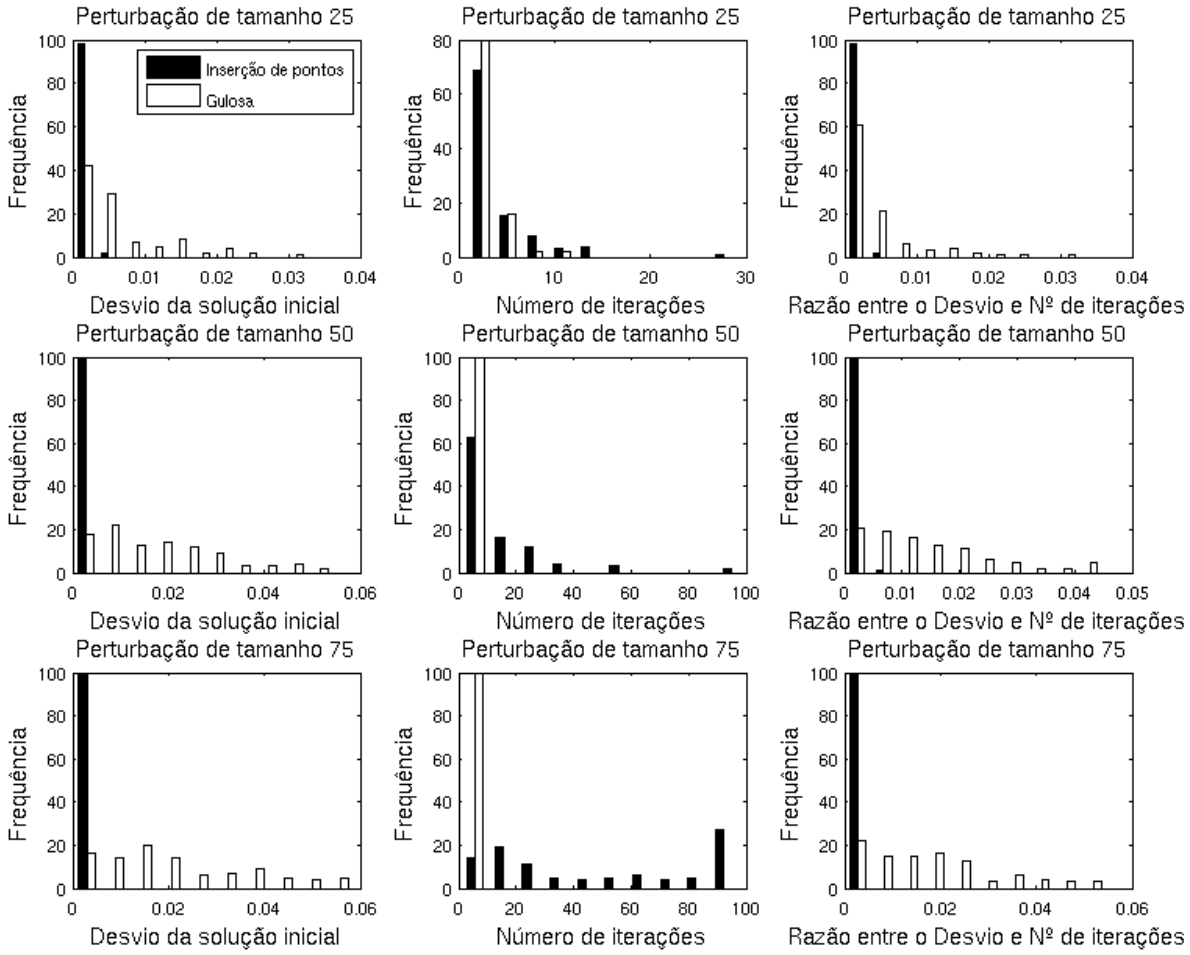

Figura A.8: Continuação da FiguraA.7 - Histogramas para a calibração do tamanho da perturbação para instâncias com 100 pontos dados.

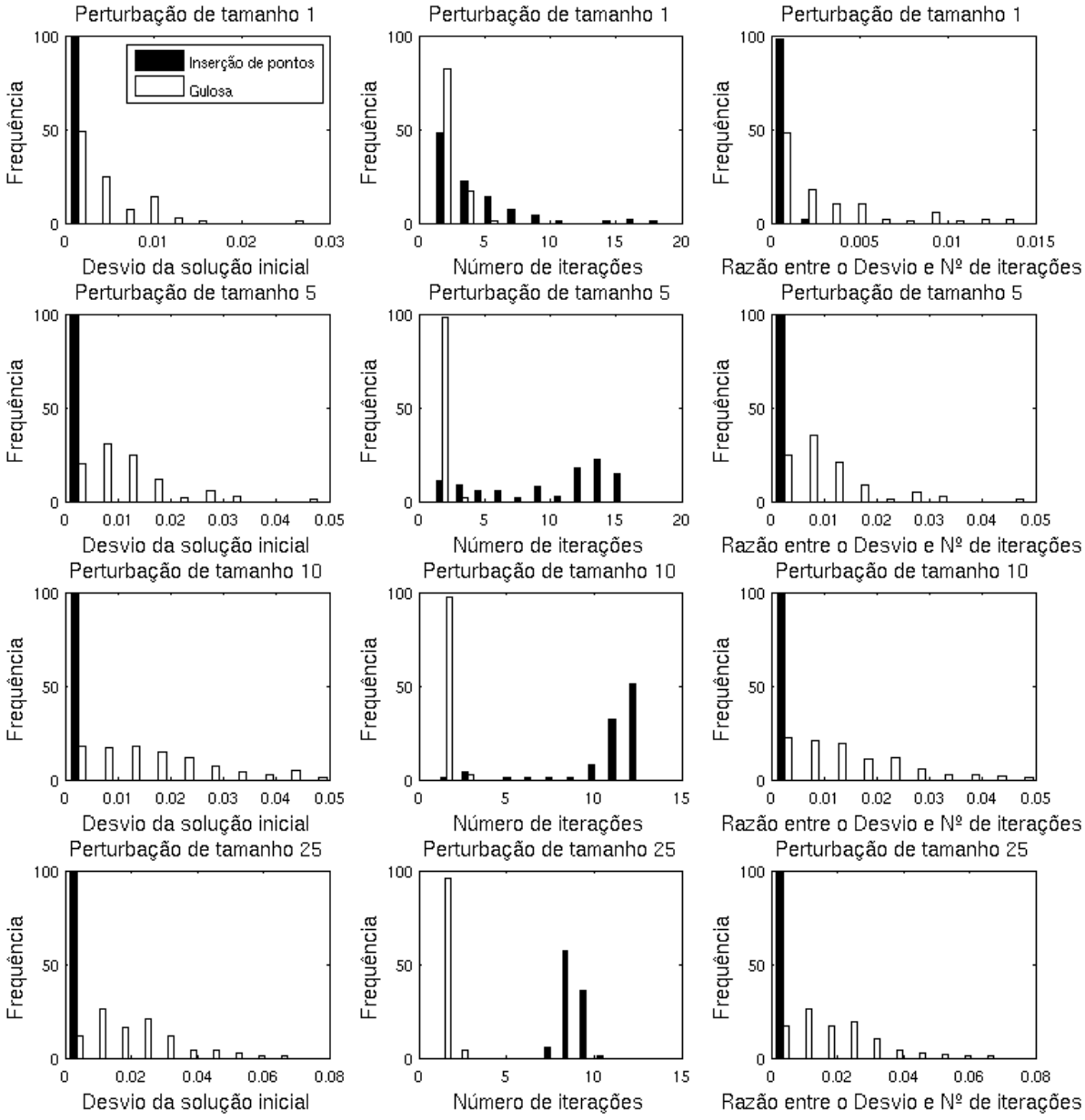

Figura A.9: Histogramas para a calibração do tamanho da perturbação para instâncias com 250 pontos dados.

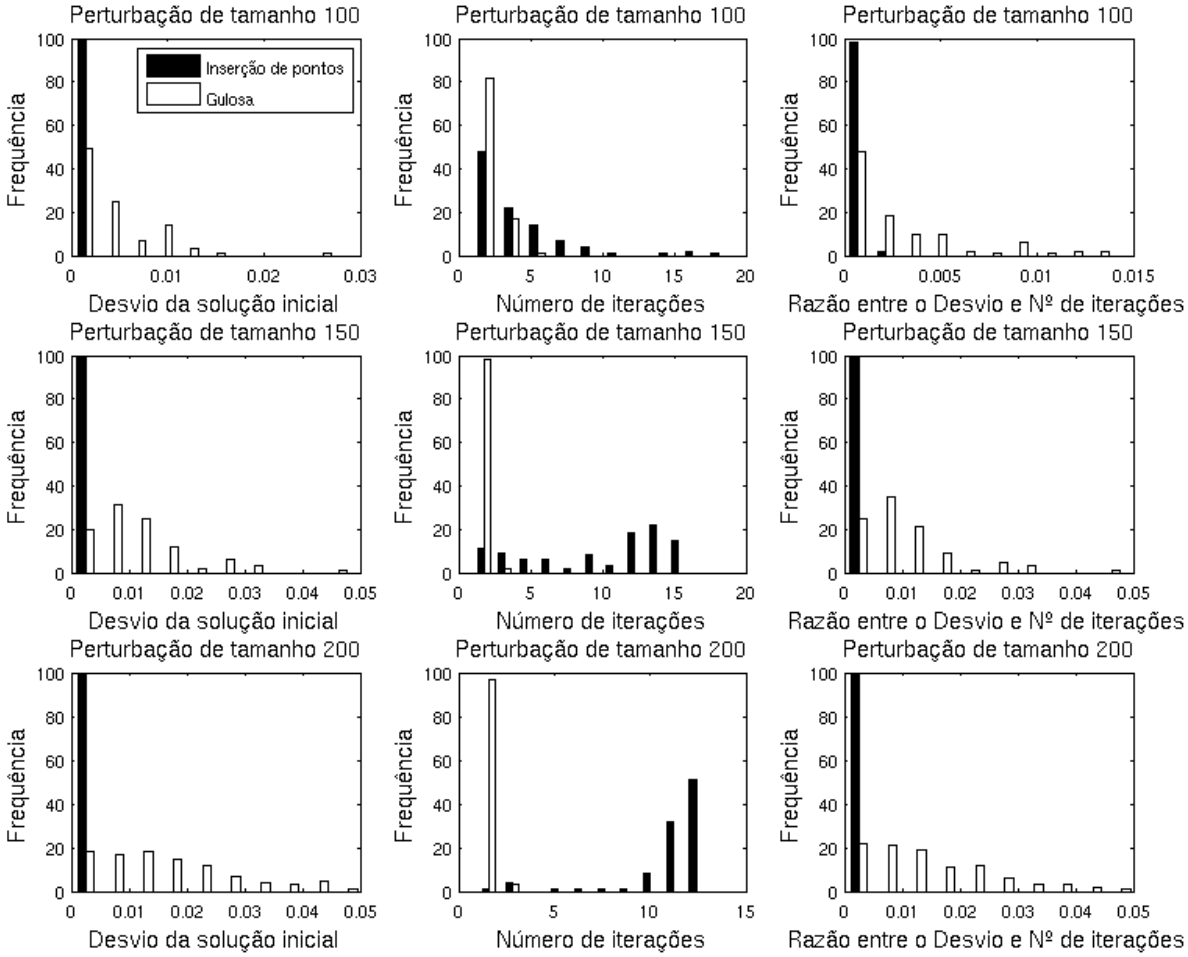

Figura A.10: Continuação da FiguraA.9 - Histogramas para a calibração do tamanho da perturbação para instâncias com 250 pontos dados.