Uma Abordagem de Programação Inteira para o Problema da Triangulação de Custo Mínimo

Aminadab Pereira Nunes

Dissertação de Mestrado

Uma Abordagem de Programação Inteira para o Problema da Triangulação de Custo Mínimo

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Aminadab Pereira Nunes e aprovada pela Banca Examinadora.

Campinas, 27 de Novembro de 1997.

Prof. Dr. Cid Carvalho de Souza (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Instituto de Computação
Universidade Estadual de Campinas

Uma Abordagem de Programação Inteira para o Problema da Triangulação de Custo Mínimo

Aminadab Pereira Nunes

Outubro de 1997

Banca Examinadora:

- Prof. Dr. Cid Carvalho de Souza (Orientador)
- Prof. Dr. Nelson Maculan
  COPPE – UFRJ
- Prof. Dr. Jorge Stolfi
  Instituto de Computação – Unicamp
- Prof. Dr. Arnaldo Vieira Moura (Suplente)
  Instituto de Computação – Unicamp

Este trabalho é dedicado aos meus pais,
a quem devo o melhor do que sou.

Agradecimentos

Ao Cid, pela oportunidade, pela excelente orientação, pela confiança, pela persistência e pela amizade. Espero, sinceramente, ter assimilado um pouco de sua competência e profissionalismo.

Aos professores do IC que, de alguma forma, contribuíram para incrementar minha formação. Em especial, ao Marcus.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — CAPES, e à Fundação de Amparo à Pesquisa do Estado de São Paulo — FAPESP (Proc. No. 95/8929-9) pelo apoio financeiro, sem o qual este trabalho não seria viável.

À Maria pelo amor, paciência e cumplicidade. Falar mais seria minimizar sua importância para mim neste período.

Aos amigos que aqui encontrei, aqueles capazes de suavizar uma trajetória nem sempre linear. De forma especial, ao Cláudio, Delano, Elder, Jerônimo, Marcos André, Nalon, Nivando e Uchôa.

Por último, justamente para destacar, à minha família. Aos meus pais, Geraldo e Maria, com quem aprendi o que um pai é capaz de fazer (abdicar, sofrer, torcer) por um filho e aos meus irmãos, Andréia, Robson e Fabrício, pelo amor e torcida. Não poderia agradecê-los o suficiente.

Acreditar na esperança dourada do sol
Mesmo que em plena boca
Nos bata o açoite
Contínuo da noite.

– *João Bosco e Aldir Blanc*

Sumário

Seja P um conjunto finito de pontos no plano e $S(P)$ o conjunto de todos os segmentos de reta com extremos em P. Uma triangulação planar de P é um subconjunto maximal de $S(P)$ tal que nenhum par de segmentos neste subconjunto se intercepta, exceto possivelmente nos extremos. Chamamos de triangulação de custo mínimo a triangulação planar cuja soma total dos comprimentos de seus segmentos de reta é mínimo dentre todas as triangulações planares de P. Não se conhece algoritmo polinomial que resolva o problema de determinar a triangulação de custo mínimo de um conjunto de pontos no caso geral, contudo, também não está provado tratar-se de um problema *NP-difícil*.

Neste trabalho estamos interessados na resolução exata deste problema. Nossa abordagem é baseada em técnicas de programação inteira, em particular estudamos duas formulações distintas para o problema. A primeira formulação é baseada em uma equivalência entre o problema da triangulação de custo mínimo e uma versão restrita do problema do conjunto independente em um grafo. Além das desigualdades obtidas através da observação desta equivalência, mostramos como fortalecer a formulação através de certas propriedades geométricas do problema. Estudamos ainda uma outra formulação baseada principalmente no trabalho apresentado por Loera et. al em [dLHSS96]. Enquanto na primeira formulação as variáveis binárias estão associadas aos segmentos em $S(P)$, nesta segunda formulação as variáveis binárias estão associadas aos triângulos com vértices em P. Os resultados computacionais que obtivemos mostram uma clara superioridade do segundo modelo. Para a primeira formulação implementamos um algoritmo *branch-and-cut* que nos permitiu resolver problemas de até 160 pontos ($|P| = 160$). Já para a segunda formulação a solução ótima da relaxação linear sempre foi inteira, o que nos permitiu resolver instâncias com até 1000 pontos ($|P| = 1000$).

Abstract

Let P be a finite set of points in the plane and $S(P)$ be the set of all segments with both extreme points in P. A planar triangulation of P is a maximal subset of $S(P)$ such that no pair of segments is this subset intercept each other, except possibly at their extremities. A minimum triangulation of P is a planar triangulation whose sum of the lengths of all its segments is minimum over all possible triangulations of P. No polynomial algorithm is known that solves this problem in the general case, however it is also not known if the problem is *NP-hard*.

In this work we are interested in solving the problem exactly. Our approach is based on integer programming techniques and is particular we have studied two different formulations for the problem. The first formulation is based on an equivalence between the problem of finding a minimum weight triangulation of P and a restricted version of the maximum independent set of a graph. Besides the inequalities arising from this observation, we show how to strength the formulation by using geometric properties if the problem. We also have studied a second formulation mainly based on the work of Loera et. al [dLHSS96]. While in the first formulation the binary variables are associated to the segments in $S(P)$, in this second formulation the binary variables are associated tho the triangles with vertices lying in P. Our computational results have shown that the second model clearly outperforms the first one. For the first formulation, we have implemented a branch-and-cut algorithm which allowed us to solve instances with up to 160 points ($|P| = 160$). On the other hand, for the for second formulation, the optimal solution of the linear relaxation was integer for all tested instances, which has made possible the the solution of instances with up to 1000 points ($|P| = 1000$).

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Geometria computacional é o ramo da ciência da computação que estuda métodos algorítmicos para resolução de problemas geométricos. Estes problemas são tipicamente definidos sobre objetos geométricos tais como conjuntos de pontos, conjuntos de segmentos de reta, ou polígonos. Em geral, o objetivo ou consiste em responder questões referentes a estes objetos, tal como se existe interseção entre segmentos de um certo conjunto, ou em identificar um novo objeto geométrico, tal como a envoltória convexa de um certo conjunto de pontos.

Nesta dissertação tratamos do problema de geometria computacional conhecido como *problema da triangulação de custo mínimo* (PTCM). Seja P um conjunto finito de pontos no plano. Uma *triangulação planar* de P é definida como um conjunto maximal de segmentos de reta conectando estes pontos, tal que quaisquer dois destes segmentos não se interceptam, exceto possivelmente nos extremos. O problema em questão consiste em determinar uma triangulação planar que minimize a soma dos comprimentos de seus segmentos de reta. Uma triangulação planar que possua esta propriedade é chamada de *triangulação de custo mínimo* (TCM). Na Figura 1.1 vemos duas triangulações planares do mesmo conjunto de pontos, sendo que a da esquerda é uma TCM. Como pode ser observado nesta figura, a condição de maximalidade do número de segmentos impõe que todas as regiões limitadas formadas sejam triângulos, daí o nome triangulação planar.

A TCM encontra sua principal aplicação na aproximação numérica de uma função de duas variáveis. Considere a função $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Seja P um conjunto finito de pontos em $\mathbb{R}^2$, para os quais o valor de f é conhecido. Em [Yoe75], Yoeli propõe uma abordagem chamada método poliedral para calcular uma aproximação do valor de f para qualquer ponto $q \notin P$ contido na envoltória convexa de P. Basicamente, este método consiste em discretizar a envoltória convexa de P em triângulos cujos extremos estão em P (construir

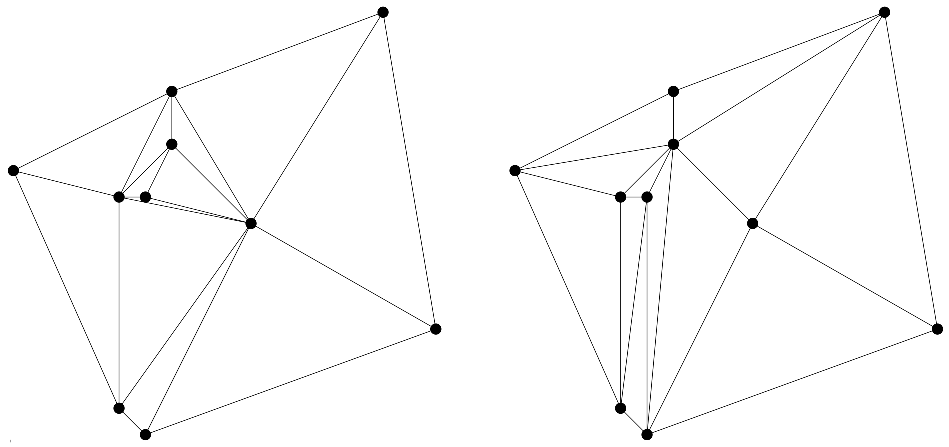


Figura 1.1: Dois exemplos de triangulações planares.

uma triangulação planar) e então calcular a aproximação para o valor de $f(q)$ através da interpolação linear dos valores de f para os vértices do triângulo no qual o ponto q está contido. Yoeli argumenta que, na representação de superfícies, onde $f(q)$ representa a altura do ponto q, a TCM apresenta propriedades que provêm uma boa aproximação para esta função.

Além da aplicação mencionada acima, a TCM apresenta ainda outras propriedades proveitosas. Destaca-se dentre estas o fato de que a TCM pode representar uma boa aproximação de um grafo completo. Um grafo completo representa uma rede de comunicação ideal entre n localidades. Entretanto, geralmente, visando poupar recursos, redes esparsas são projetadas de forma a aproximarem-se de tais grafos em algum sentido. No caso onde os grafos são euclideanos (vértices equivalem a pontos no plano e arestas a segmentos de reta entre estes pontos) é comum projetar-se tais aproximações de forma que a menor distância entre quaisquer duas destas localidades seja limitada por uma constante múltipla de sua separação linear (a menor distância entre estas localidades). Em [DJ89], Das e Joseph mostram que a TCM pode ser utilizada como uma aproximação de um grafo completo neste sentido.

Do ponto de vista teórico o PTCM se destaca por representar um desafio aos pesquisadores de geometria computacional. A complexidade deste problema permanece em aberto por mais de duas décadas [GJ79]. Ou seja, apesar de não se conhecer algoritmos determinísticos polinomiais que o resolva não se conseguiu provar tratar-se de um problema *NP-difícil*. Além disto, até recentemente não existia sequer prova de que algum dos métodos aproximados utilizados em sua resolução fosse capaz de fornecer alguma garantia de qualidade.

Tradicionalmente, o PTCM tem sido abordado na literatura de forma aproximada, onde os métodos de resolução propostos não necessariamente conduzem à solução ótima. Contudo, o sucesso obtido por métodos recentes na resolução exata de certas instâncias do problema têm motivado novos trabalhos que abordam o tema.

Nossa meta nesta dissertação é verificar a adequabilidade do uso de técnicas de *programação inteira* (PI) à resolução exata do PTCM. Neste sentido avaliamos dois modelos de PI para o problema.

A utilização de dois modelos é baseada na observação de que uma TCM pode ser definida tanto como um conjunto de segmentos como um conjunto de triângulos. Assim, no primeiro modelo (discutido no Capítulo 3), o qual chamamos de *modelo baseado em segmentos*, consideramos o caso onde a meta é determinar os segmentos de uma TCM. Enquanto, no segundo modelo (discutido no Capítulo 4), o qual chamamos de *modelo baseado em triângulos*, consideramos o caso onde o objetivo é identificar os triângulos que compõem uma TCM.

Para resolver o problema utilizando o modelo baseado em segmentos optamos por um algoritmo baseado em planos-de-corte, o *branch-and-cut*. Algoritmos *branch-and-cut* têm sido utilizados com sucesso na resolução de instâncias práticas de problemas combinatórios notoriamente difíceis, tal como o *problema do caixeiro viajante*. Contudo, a aplicação desta técnica requer uma análise detalhada da estrutura do problema, especificamente, requer o conhecimento de desigualdades válidas fortes para a envoltória convexa (politopo) das soluções viáveis do problema de PI associado.

Desta forma, no Capítulo 3, mostramos como formular a versão sobre segmentos do PTCM como um problema de PI e como fortalecer este modelo através de outras desigualdades válidas fortes. Mostramos ainda que algumas destas desigualdades, ou casos particulares delas, definem facetas no politopo monótono. Finalmente, no Capítulo 5 apresentamos rotinas capazes de gerar tais desigualdades automaticamente.

Na resolução do problema utilizando o modelo baseado em triângulos, o qual foi fundamentalmente baseado no trabalho de Loera et. al apresentado em [dLHSS96], surpreendentemente, para as intâncias testadas, não foi necessário resolver mais que uma relaxação linear para identificarmos uma solução ótima do problema. Isto porque a solução ótima da relaxação linear sempre foi inteira, i.e., a solução da relaxação sempre coincidiu com uma solução do problema original. Na realidade, o politopo definido pela relaxação linear não corresponde, no caso geral, à envoltória convexa das soluções viáveis, entretanto, nas instâncias testadas sempre houve uma solução correspondente a alguma TCM que fosse vértice deste politopo.

Do ponto de vista computacional, o grande problema apresentado pelo modelo baseado em triângulos é seu elevado número de variáveis. Visando contornar este problema, investigamos a viabilidade da utilização de um algoritmo de *geração de colunas*, o qual é capaz de resolver o problema sem considerar todas as suas variáveis simultaneamente.

O restante deste capítulo é dedicado à apresentação da terminologia, conceitos e resultados básicos a serem utilizados no decorrer desta dissertação. Na próxima seção apresentamos algumas definições básicas de geometria. Na Seção 1.2 caracterizamos um problema de PI e apresentamos os principais algoritmos utilizados para resolvê-los à otimalidade. Visto que a eficiência de um *branch-and-cut* depende da força das desigualdades utilizadas, apresentamos na Seção 1.3 alguns resultados básicos de teoria poliedral os quais nos fornecem as ferramentas necessárias para avaliar teoricamente a força das desigualdades apresentadas. Finalmente, na Seção 1.4 apresentamos a organização do restante desta dissertação.

1.1 Conceitos Básicos de Geometria

Nesta seção apresentamos algumas definições e convenções básicas envolvendo objetos geométricos a serem utilizadas no restante desta dissertação. As definições foram, em sua maioria, compiladas de [PS85, Tan93].

Disposição geral. Seja P um conjunto de $n \geq 3$ pontos no plano. Dizemos que P está em disposição geral caso não possua 3 pontos colineares.

O estudo de um problema de geometria computacional definido sobre um conjunto de pontos pode ser significativamente dificultado se considerarmos que estes pontos podem não estar em disposição geral. Observe que, o conceito de triangulação planar sequer se aplica a um conjunto de pontos colineares. Assim, por simplicidade, a menos que seja dito contrário, todo conjunto finito de pontos considerado neste texto está em disposição geral.

Segmento de reta. Dados dois pontos $p_1, p_2 \in I\!R^d$, uma *combinação linear convexa* destes pontos é dada por qualquer ponto p_3 obtido através da seguinte expressão:

$$\alpha p_1 + (1-\alpha)p_2, \quad \alpha \in I\!R,\ 0 \leq \alpha \leq 1.$$

O segmento de reta $\overline{p_1 p_2}$ representa o conjunto de possíveis combinações convexas de p_1 e p_2. Para os nossos propósitos, diremos que dois segmentos se interceptam apenas se esta interseção não se der em seus extremos.

Polilinha. Em $I\!R^2$ uma polilinha é caracterizada por uma seqüência de segmentos de reta $S = \{s_1, s_2, ..., s_n\}$, onde todo segmento da seqüência compartilha um de seus vértices apenas com o segmento que o antecede na lista. Caso um dos extremos de s_1 coincida com um dos extremos de s_n dizemos que a polilinha é *fechada*, caso contrário dizemos que é *aberta*.

Conjunto convexo. Dizemos que um domínio D em $\mathbb{R}^d$ é *convexo* se para quaisquer dois pontos $p, q \in D$ o segmento de reta $\overline{pq}$ está inteiramente contido em D.

Região. Designamos por região qualquer conjunto de pontos que componha um dos elementos de uma dada partição de $\mathbb{R}^2$.

Polígono. Um polígono consiste em uma polilinha fechada. Um polígono é dito ser *simples* se nenhum par de seus segmentos se interceptam. Um polígono simples particiona o plano em duas regiões distintas, o *interior* (limitado) e o *exterior* (não-limitado). Os pontos sobre os segmentos que definem um polígono simples pertencem ao interior deste polígono.

Grafo euclideano. Seja $G = (V, E)$ um grafo simples, onde V representa seu conjunto de vértices e E seu conjunto de arestas. Dizemos que G é um grafo *euclideano* caso seus vértices representem pontos em $\mathbb{R}^2$, e suas arestas segmentos de reta entre estes pontos. Um grafo euclideano é dito ser *planar* se não há interseções entre quaisquer dois de seus segmentos.

Triangulação planar. Seja P um conjunto finito de $n \geq 3$ pontos no plano. Uma triangulação planar de P é um conjunto maximal de segmentos de reta ligando pontos de P, tal que quaisquer dois destes segmentos não se interceptam.

A maximalidade do número de segmentos em uma triangulação planar impõe com que todas as regiões limitadas formadas sejam *triângulos vazios* com extremos em P, onde entende-se por triângulo vazio qualquer triângulo que não contenha pontos de P no seu interior, exceto possivelmente os extremos. Este fato possibilita definir alternativamente uma triangulação planar como um conjunto maximal de triângulos vazios com extremos em P, tal que o conjunto de pontos na interseção dos interiores de quaisquer dois destes triângulos não tenha dimensão maior que 1.

Triangulação de custo mínimo. Dado um conjunto de pontos P uma triangulação de custo mínimo deste conjunto é uma triangulação planar cuja soma dos comprimentos (distância euclideana) de seus segmentos de reta é mínima dentre todas as triangulações planares de P.

Dado um conjunto finito P de $n \geq 3$ pontos no plano, no restante deste texto, também será comum a referência aos seguintes conjuntos:

$S(P)$**:** Conjunto de segmentos de reta com extremos em P.

$\triangle(P)$: Conjunto de triângulos vazios com extremos em P.

1.2 Conceitos Básicos de Programação Inteira

Um problema de *programação linear* (PL) consiste em otimizar uma função linear (função objetivo) sobre uma região descrita por um conjunto de desigualdades lineares (restrições do problema). Os pontos no interior desta região formam o conjunto de soluções viáveis para o problema. Isto pode ser representado de forma matricial como:

$$\begin{array}{ll} \min & cx \\ \text{Sujeito a} & Ax \leq b \\ & x \in I\!\!R^n_+ \end{array} \tag{1.1}$$

onde $c \in I\!\!R^n$, A é uma matriz $m \times n$ e $b \in I\!\!R^m$. Se as variáveis x devem ser restritas a valores inteiros ($x \in Z\!\!\!Z^n_+$, ao invés de $x \in I\!\!R^n_+$), o problema é chamado de problema de programação linear inteira, ou simplesmente, programação inteira (PI). Além disso, se variáveis do problema são restritas aos valores em $\{0, 1\}$, teremos um problema de PI 0-1.

Apesar da diferença entre PL e PI parecer sutil, via de regra, ela torna os problemas da segunda classe extremamente mais difíceis. Problemas de PL podem ser resolvidos em tempo polinomial [Sch86], enquanto a grande maioria de problemas de PI são *NP-difíceis*.

A forma mais usual de se tentar resolver um problema de PI é através de sua relaxação linear. A *relaxação linear* de um problema de PI é o problema de PL correspondente, onde são eliminadas as restrições de integralidade, i.e., onde a restrição $x \in Z\!\!\!Z^n_+$ é substituída pela restrição $x \in I\!\!R^n_+$. Exemplos de algoritmos clássicos que utilizam esta abordagem para resolução de problemas de PI são os algoritmos *cutting-planes* (de planos-de-cortes) e *branch-and-bound* (de enumeração implícita).

Para que se possa compreender como estes algoritmos utilizam a relaxação linear de um problema PI para resolvê-lo, apresentamos na seqüência alguns resultados básicos de PL.

Definições e Resultados Básicos de PL

Os teoremas e definições abaixo foram compilados de [NW88, Sch86]. Considere o problema genérico de PL (1.1):

Teorema 1.1 *O conjunto de soluções viáveis* $Q = \{x : Ax \leq b, x \in I\!\!R^n_+\}$ *para o problema é um conjunto convexo. Ou seja, a combinação linear convexa de quaisquer dois pontos* *em Q resulta em um terceiro ponto que também está contido em Q.*

Definição 1.1 *O conjunto convexo $Q = \{x : Ax \leq b, x \in I\!R^n_+\}$ é denominado* poliedro. *Se Q é limitado, i.e., $Q \subseteq \{x : -w \leq x_j \leq w, \forall\ j \in \{1, 2, \cdots, n\}\}$ para algum $w \in I\!R_+$, então Q é chamado de* politopo.

Definição 1.2 *Um* vértice *de um poliedro Q é qualquer ponto $x \in Q$ o qual não pode ser expresso como uma combinação linear convexa de outros pontos de $Q \setminus \{x\}$.*

Teorema 1.2 *Se o valor ótimo de uma função linear em um poliedro $Q \subseteq I\!R^n$ é finito, então ele é atingido em pelo menos um vértice. Se este for obtido em mais que um vértice, então pode ser obtido também por qualquer ponto que seja uma combinação linear convexa destes vértices.*

Teorema 1.3 *Um problema de PL onde os elementos da matriz A são racionais pode ser resolvido em tempo polinomial sobre n, m e θ, onde n é o número de variáveis do problema, m é o número de restrições, e θ é tamanho da maior representação binária de um coeficiente da matriz A.*

1.2.1 Algoritmos Utilizados na Resolução de Problemas de PI

Nesta seção caracterizamos os principais algoritmos empregados na resolução de problemas de PI que se utilizam de relaxação lineares. Dentre estes destacam-se o algoritmo *branch-and-cut* e o algoritmo de geração de colunas, os quais empregamos na resolução do PTCM.

Algoritmo Branch-and-Bound

Supondo que o espaço de soluções viáveis é limitado, o que é verdade no caso de problemas de PI 0-1, uma idéia intuitiva, apesar de ingênua, seria enumerar estas soluções a fim de identificarmos uma que otimize a função objetivo em questão. Obviamente, este algoritmo termina em tempo finito, porém o tempo despendido nesta operação é linear no tamanho do espaço de soluções, o qual pode ser exponencial no número de variáveis do problema.

A idéia do algoritmo *branch-and-bound* é enumerar todas as soluções viáveis de forma implícita. Para isso, particiona-se o conjunto de soluções em subconjuntos disjuntos em uma tentativa de, devido às propriedades desses subconjuntos, se caracterizar quais subconjuntos de soluções não podem conter uma solução ótima. Desta forma, procura-se evitar a enumeração explícita de todas as soluções.

Usualmente, a partição dos conjuntos de soluções é construída de forma recursiva, permitindo-se assim que o processo seja visualizado como uma árvore: a árvore de enumeração. Nesta árvore, os filhos de um nó correspondem a uma partição do conjunto que o nó representa. Assim a raiz representa o conjunto de todas as soluções.

Consideremos um problema de PI 0-1 de minimização e o conjunto de todas as suas soluções viáveis $\mathcal{X}$. Cada nó i na árvore de enumeração representa uma relaxação linear de um problema de PI cujas soluções viáveis são $\mathcal{X}^i \subseteq \mathcal{X}$. Seja z_i^* o valor ótimo desta relaxação, dependendo do valor de z_i^* o nó i pode dar origem a dois outros nós (seus filhos) ou pode ser podado, i.e., ou particiona-se o subconjunto $\mathcal{X}^i$ em dois novos subconjuntos, ou ele não será particionado até o fim da execução do algoritmo.

Caso o algoritmo decida-se por particionar (*branching*) o subconjunto $\mathcal{X}^i$, uma variável fracionária x_i é escolhida. Então, os dois filhos do nó i passarão a representar relaxações lineares dos problemas de PI cujas soluções viáveis são $\mathcal{X}_0^i$ e $\mathcal{X}_1^i$, onde $\mathcal{X}_0^i = \{x : x \in \mathcal{X}^i, x_i = 0\}$ e $\mathcal{X}_1^i = \{x : x \in \mathcal{X}^i, x_i = 1\}$. Ou seja, um dos filhos passará a representar a relaxação linear de seu pai acrescida da restrição $x_i = 0$, e o outro passará a representar a relaxação linear de seu pai acrescida da restrição $x_i = 1$.

É fácil observar que o valor ótimo da relaxação linear de um filho não pode ser menor que a de seu pai (num problema de minimização). Assim, conhecendo-se um limite superior para a solução ótima do problema de PI original, o qual pode ser obtido, p.ex., através de uma heurística, pode-se podar qualquer nó cujo valor da relaxação linear for maior que o limite superior em questão (*bounding*). Além disso, nós cuja relaxação linear fornecem uma solução inteira também podem ser podados. Neste último caso, se esta solução fornece um limite superior melhor que o limite utilizado no momento, então atualiza-se o limite superior corrente.

Num dado momento da execução do algoritmo, os nós da árvore que não estão podados são ditos serem nós ativos. O algoritmo termina quando não existem mais nós ativos, e a solução ótima é a solução inteira de menor custo obtida no processo.

Algoritmo Cutting-Planes

Algoritmos *cutting-planes* são baseados no conceito de fortalecimento da relaxação linear através de sucessivas inserções de *desigualdades válidas*. Uma desigualdade é dita ser válida com respeito a um certo conjunto de pontos, se todos os pontos deste conjunto a satisfazem.

Do Teorema 1.2 temos que, uma solução ótima de um problema de PL pode ser encontrada em um dos vértices do poliedro. A idéia básica dos algoritmos *cutting-planes* é "lapidar" o poliedro definido pela relaxação linear do problema de PI original (através da inserção de novas desigualdades válidas — planos de corte) a fim de fazer com que este se aproxime da *envoltória convexa* das soluções inteiras, i.e., do menor poliedro que contém estas soluções. Assim teríamos o ótimo inteiro como um vértice deste novo poliedro e através da resolução de uma relaxação linear poderíamos identificá-lo.

De forma sucinta o funcionamento deste tipo de algoritmo é o seguinte: começa-se resolvendo uma relaxação linear do problema de PI em questão. Se a solução ótima obtida é um ponto em coordenadas inteiras, então o problema está resolvido. Caso contrário, o algoritmo deve inserir uma nova desigualdade ao problema, tal que esta deve eliminar o vértice previamente obtido sem, contudo, eliminar qualquer solução viável (solução inteira). O algoritmo repete o processo até que um ótimo inteiro seja encontrado. A Figura 1.2 [dS93] exibe graficamente esta idéia sobre um problema de maximização. Os círculos representam os pontos em coordenadas inteiras. As linhas cheias representam as desigualdades que determinam o poliedro inicial (relaxação linear). As soluções viáveis são representadas por círculos preenchidos. A seta indica a direção da função objetivo, e a linha tracejada representa uma desigualdade válida que elimina a solução ótima da relaxação linear.

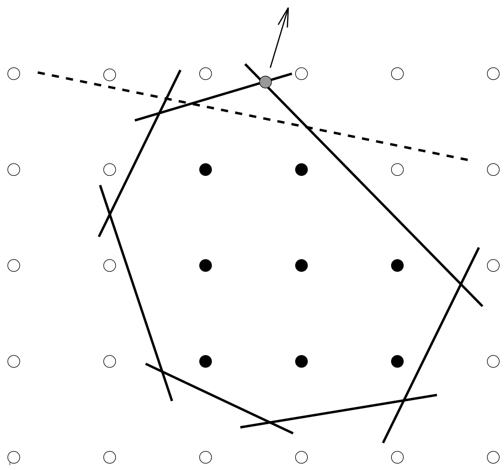


Figura 1.2: Esboço do funcionamento de um algoritmo *cutting-planes*.

Um estudo inicial da adição de desigualdades válidas para problemas genéricos de PI foi feito por Gomory na década de 50. Infelizmente, as desigualdades propostas por ele não se mostraram eficientes na prática pois o algoritmo, embora convergindo em um tempo finito, é muito lento. No entanto, recentemente, o interesse pelos cortes de Gomory ressurgiu com a publicação de um artigo de Balas et. al [BCCN96], que parece indicar uma nova possibilidade para utilização destes cortes.

Algoritmo Branch-and-Cut

Suponha que em um dado momento da execução de um algoritmo *cutting-planes* o problema de PL corrente defina um poliedro Q, no qual toda solução ótima é fracionária, e que duas desigualdades válidas $\pi x \leq \pi_0$ e $\lambda x \leq \lambda_0$ podem ser utilizadas como corte. Suponha também que $(\{x : \pi x \leq \pi_0\} \cap Q) \subset (\{x : \lambda x \leq \lambda_0\} \cap Q)$. Não é difícil perceber que a utilização de $\pi x \leq \pi_0$ como corte dispensa a utilização de $\lambda x \leq \lambda_0$ e a recíproca não é verdadeira (o corte que $\pi x \leq \pi_0$ define é mais "profundo"). Nesta situação, dizemos que $\pi x \leq \pi_0$ *domina*, ou é *mais forte* que, $\lambda x \leq \lambda_0$.

Como a eficiência prática de um algoritmo *cutting-planes* aparenta estar intimamente ligada com a quantidade de cortes que este deve gerar para solucionar o problema, parece intuitivo que quanto mais fortes forem as desigualdades que o algoritmo utiliza como corte, menor será o número de cortes necessários e maior a probabilidade de que o algoritmo seja computacionalmente mais eficiente.

Um método relativamente recente que considera este conceito de força de desigualdades é o algoritmo *branch-and-cut*. Basicamente, este algoritmo consiste em um algoritmo *branch-and-bound* onde a cada nó da árvore de enumeração executa-se um algoritmo *cutting-planes*, o qual utiliza apenas desigualdades válidas fortes. Seja $\mathcal{X}$ o conjunto de soluções viáveis de um problema de PI e seja $conv(\mathcal{X})$ sua envoltória convexa. Considere também um conjunto $\mathcal{F}$ de desigualdades fortes com respeito a $conv(\mathcal{X})$ e uma rotina, chamada *rotina de separação*, capaz de determinar rapidamente uma desigualdade em $\mathcal{F}$ que é violada por um ponto $x \notin conv(\mathcal{X})$. Podemos com isto descrever um algoritmo de *branch-and-cut*.

Considere uma iteração onde o nó i da árvore de enumeração está sendo analisado. Seja x^i a solução ótima obtida com a relaxação linear. Se x^i contém elementos fracionários, utiliza-se de uma rotina de separação para encontrar uma nova desigualdade válida em $\mathcal{F}$ a qual x^i não satisfaça. Se a rotina obtém sucesso a nova desigualdade é adicionada à relaxação linear (*cutting-planes*), e reinicia-se o processo. Este ciclo é quebrado quando: $z_i = cx^i$ for maior que o limite superior corrente, ou $x^i \in \mathbb{Z}^n$, ou se a rotina de separação não conseguir encontrar uma nova desigualdade válida violada por x^i. Neste último caso, uma variável fracionária é escolhida e faz-se um *branching* sobre ela.

A diferença fundamental entre *branch-and-cut* e outros métodos *cutting-planes* é a utilização apenas de desigualdades fortes. Note que estas não compõem uma classe genérica de desigualdades, pois dependem da estrutura da envoltória convexa das soluções viáveis do problema em questão. As principais tarefas envolvidas no desenvolvimento deste tipo de algoritmo é caracterizar quais são estas desigualdades e determinar como elas podem ser computadas eficientemente por uma rotina de separação.

Algoritmo de Geração de Colunas

Não é raro encontramos modelos de PL cujo elevado número de variáveis dificulta, ou mesmo impossibilita, sua resolução computacional. Uma alternativa usual para solucionar esta classe de problemas é a utilização de algoritmos de *geração de colunas*. Em princípio, tais algoritmos são capazes de resolver um problema de PL sem considerar todas as suas variáveis simultaneamente.

A idéia básica do funcionamento de um algoritmo de geração de colunas é bastante simples. Inicialmente consideramos apenas parte das variáveis do problema original. Então, a cada passo da execução do algoritmo resolvemos o problema de PL corrente e, através da análise dos custos reduzidos das variáveis não consideradas neste problema, verificamos se a inserção de algumas destas pode melhorar o valor da solução ótima corrente. Em caso afirmativo, tais variáveis, ou parte delas, são inseridas no problema de PL corrente (novas colunas da matriz de restrições são geradas) e repete-se o processo. Em caso negativo, temos uma garantia de que a solução ótima do problema corrente é a solução ótima do problema original, e o algoritmo pode parar. Tal como descrito, no pior caso o algoritmo necessitará inserir todas as variáveis desconsideradas no início, contudo, na prática, é pouco provável que isto aconteça.

Quando, ao invés de um problema de PL, visamos resolver computacionalmente problemas de PI com elevado número de variáveis, onde a simples utilização da relaxação linear não é capaz de resolver o problema, é comum associarmos o algoritmo de geração de colunas com um algoritmo *branch-and-bound*. O algoritmo resultante desta associação é costumeiramente chamado de *branch-and-price*. A grosso modo, este algoritmo consiste de um algoritmo *branch-and-bound* onde utilizamos um algoritmo de geração de colunas para resolver as relaxações lineares surgidas em cada um dos nós da árvore de enumeração.

1.3 Conceitos Básicos de Teoria Poliedral

Nesta seção apresentamos algumas definições e resultados básicos de teoria poliedral a serem utilizados no transcorrer desta dissertação. Inicialmente formalizamos conceitos tais como dimensão, face e faceta. Em seguida, mostramos como caracterizar facetas. Finalmente, apresentamos a relação entre um problema de otimização e um problema de separação.

1.3.1 Definições Básicas

As seguintes definições foram compiladas de [NW88, dS93, FW96].

Definição 1.3 *Um conjunto de pontos* $x^1, ..., x^k \in \mathbb{R}^n$ *é* linearmente independente *se a única solução para* $\sum_{i=1}^{k} \lambda_i x^i = 0$ *é* $\lambda_i = 0$ *para todo* $i \in \{1, ..., k\}$.

Definição 1.4 *Um conjunto de pontos* $x^1, ..., x^k \in \mathbb{R}^n$ *é* afim independente *se a única solução para* $\sum_{i=1}^{k} \lambda_i x^i = 0$, $\sum_{i=1}^{k} \lambda_i = 0$ *é* $\lambda_i = 0$ *para todo* $i \in \{1, ..., k\}$.

Note que, se um conjunto de pontos $x^1, ..., x^k \in \mathbb{R}^n$ é linearmente independente, então também é afim independente.

Definição 1.5 *Seja* $P = \{x^1, x^2, ..., x^k\}$ *um conjunto de pontos em* $\mathbb{R}^n$. *A* envoltória convexa *de* P *é o conjunto de pontos dado por*

$$\text{conv}\,(P) = \{\sum_{i=1}^{k} \lambda_i x^i : \sum_{i=1}^{k} \lambda_i = 1, x^i \in P, \lambda_i \in \mathbb{R}_+, i = 1, ..., k\}.$$

Definição 1.6 *Um poliedro* $Q \subseteq \mathbb{R}^n$ *é dito ser* monótono *se e somente se* $x \in Q$ *implica* $Q \supseteq \{y \in \mathbb{R}^n_+ : y \leq x\}$.

No decorrer deste texto quando estiver claro o problema de PI do qual estamos tratando, o termo *poliedro monótono* será utilizado para designar o menor poliedro monótono que contém a envoltória convexa das soluções deste problema.

Definição 1.7 *Uma desigualdade* $\pi x \leq \pi_0$ *é* válida *para um poliedro se todos os pontos deste poliedro a satisfazem.*

Definição 1.8 *Seja* $\pi x \leq \pi_0$ *uma desigualdade válida para um poliedro* Q. *O conjunto* $F = \{x \in Q : \pi x = \pi_0\}$ *é chamado de* face *de* P *(dizemos que a desigualdade* $\pi x \leq \pi_0$ *define a face* F *em* Q*). Uma face* F *é dita ser* própria *se* $F \neq \emptyset$ *e* $F \neq Q$.

Definição 1.9 *Um conjunto* $D \subseteq \mathbb{R}^n$ *possui* dimensão k, *denotada por* $\dim(D) = k$, *se a cardinalidade de todo subconjunto maximal de vetores afim independentes em* D *é* $k+1$. *Caso* $\dim(D) = \dim(\mathbb{R}^n) = n$, *dizemos que* D *possui* dimensão plena.

Definição 1.10 *Seja F uma face de um poliedro $Q \subseteq I\!R^n$. Se* $\dim(F) = \dim(Q) - 1$ *então F é dita ser uma* faceta *de Q.*

Explicamos agora o que entendemos por *lifting* de uma desigualdade. Seja $\pi x \leq \pi_0$ uma desigualdade válida com respeito a um certo poliedro $Q \subseteq I\!R^n$ e seja $F_{(\pi,\pi_0)}$ a face que ela define em Q. Suponha que existe uma outra desigualdade $\gamma x \leq \gamma_0$ válida com respeito a Q, a qual define a face $F_{(\gamma,\gamma_0)}$ em Q. Dizemos que a desigualdade $\gamma x \leq \gamma_0$ é um *lifting* da desigualdade $\pi x \leq \pi_0$ se:

(i) $F_{(\pi,\pi_0)} \subset F_{(\gamma,\gamma_0)}$,

(ii) $dim(F_{(\pi,\pi_0)}) < dim(F_{(\gamma,\gamma_0)}) \leq dim(Q) - 1$.

Note que, se uma desigualdade define faceta em Q, então não pode existir uma outra desigualdade válida com respeito a Q que seja um *lifting* dela.

1.3.2 Caracterizando Facetas

Dois são os métodos usuais utilizados para se caracterizar inequações válidas que definem facetas em um certo poliedro Q. O primeiro, o qual deriva da própria definição de faceta e por isso é chamado *método direto*, consiste em mostrar a existência de exatamente $dim(Q)$ vetores afim independentes que satisfazem a inequação na igualdade. O segundo método, o qual é chamado de *método indireto*, baseia-se no resultado abaixo.

Teorema 1.4 *Seja $(A^=, b^=)$ o conjunto de igualdades de $Q \subseteq I\!R^n$ e seja $F = \{x \in Q : \pi x = \pi_0\}$ uma face própria de P. As seguintes afirmações são equivalentes:*

(i) F é uma faceta de P.

(ii) Se $\lambda x = \lambda_0$ para todo $x \in F$, então

$$(\lambda, \lambda_0) = (\alpha\pi + uA^=, \alpha\pi_0 + ub^=)$$

para algum $\alpha \in I\!R_+$ e algum $u \in I\!R^{|A^=|}$.

A utilização do método direto na caracterização de desigualdades que definem facetas, via de regra, só é viável quando lidamos com desigualdades estruturalmente muito simples. Quando tratamos de desigualdades estruturalmente mais complexas a aplicação deste método se torna quase impraticável. Na maioria das provas encontradas na literatura de que certas inequações válidas definem faceta o método utilizado é o indireto. Contudo, a aplicação do método indireto pode apresentar alguma dificuldade quando o poliedro considerado não possui dimensão plena. No caso onde o poliedro possui dimensão plena uma faceta é definida por um único hiperplano do espaço de soluções e assim basta mostrar que as inequações $\pi x \leq \pi_0$ e $\lambda x \leq \lambda_0$ definem este mesmo hiperplano, i.e., que $\lambda x \leq \lambda_0$ é um múltiplo escalar positivo de $\pi x \leq \pi_0$. No caso oposto, uma faceta pode ser definida por um número infinito de hiperplanos e desta forma devemos mostrar que o hiperplano definido por $\pi x \leq \pi_0$ consiste em uma certa rotação do hiperplano definido por $\lambda x \leq \lambda_0$.

Em face a esta dificuldade, quando o poliedro Q, definido pela envoltória convexa do problema de PI que estamos tratando, não possui dimensão plena, é uma prática usual optarmos por estudar a estrutura facial de um poliedro de dimensão plena $\tilde{Q} \supset Q$, no qual toda solução que otimiza a função objetivo do problema de PI original é um de seus vértices. O corolário abaixo formaliza a caracterização de faceta quando o poliedro possui dimensão plena.

Corolário 1.1 *Se $Q \subseteq I\!R^n$ possui dimensão plena e $F = \{x \in Q : \pi x = \pi_0\}$ é uma face própria de P. As seguintes afirmações são equivalentes:*

(i) F é uma faceta de P.

(ii) Se $\lambda x = \lambda_0$ para todo $x \in F$, então

$$(\lambda, \lambda_0) = (\alpha\pi, \alpha\pi_0)$$

para algum $\alpha \in I\!R_+$.

Visto como caracterizar desigualdades que definem facetas em um certo poliedro, utilizamos o resto desta seção para discutir a importância desta classe de desigualdades.

Seja $\mathcal{X}$ o conjunto de soluções viáveis de um problema de PI e seja $conv(\mathcal{X})$ a envoltória convexa de $\mathcal{X}$. Uma envoltória convexa define um poliedro e assim pode ser representada através de um sistema de desigualdades lineares $Ax \leq b$. Pelo Teorema 1.3, se estas desigualdades são conhecidas, e seu número é polinomial, podemos solucionar o problema em tempo polinomial, visto que todos os vértices deste poliedro estão em coordenadas inteiras. Ocorre que, um sistema de desigualdades lineares $Ax \leq b$ que define $conv(\mathcal{X})$ é minimal se e somente se toda inequação em $Ax \leq b$ define faceta em $conv(\mathcal{X})$.

Da afirmação acima vemos que, no sentido que foi definido na Seção 1.2.1, uma desigualdade que define faceta não é dominada por nenhuma outra. De fato, teoricamente, dizemos que uma desigualdade válida é tanto mais forte quanto maior é a dimensão da face que ela define, e desta forma não existe desigualdade mais forte do que aquela que define faceta. Na prática, a utilização de desigualdades válidas fortes é o principal motivo para o sucesso apresentado pelo método *branch-and-cut* na resolução de problemas combinatórios notoriamente difíceis.

Infelizmente, via de regra, não se conhece todas as inequações que definem as facetas de $conv(\mathcal{X})$. Além disso, o número destas facetas costuma ser exponencial no tamanho da entrada do problema original. Contudo, dada uma função objetivo, para que possamos resolver o problema em tempo polinomial, através de sua relaxação linear, basta que as desigualdades que definem uma solução ótima estejam presentes na formulação de PI do problema. Isto sugere que, dada uma função objetivo, apenas certas classes de facetas são necessárias para que o problema possa ser resolvido computacionalmente.

A Figura 1.3 [dS93] ilustra a idéia de um algoritmo *cutting-planes* que utiliza apenas desigualdades que definem facetas. O exemplo é o mesmo da Figura 1.2, onde os segmentos de linha pontilhados limitam a envoltória convexa das soluções do problema ($conv(\mathcal{X})$). Na Figura 1.3(a) a solução ótima da relaxação linear, a qual é fracionária, é eliminada através da inserção de uma desigualdade que define faceta em $conv(\mathcal{X})$. Na Figura 1.3(b) a nova solução ótima da relaxação linear anteriormente fortalecida é novamente eliminada por outra desigualdade que define faceta. Finalmente, na Figura 1.3(c), apesar do poliedro definido pela nova relaxação linear não corresponder a $conv(\mathcal{X})$, a solução ótima da relaxação linear está em $\mathcal{X}$ e desta forma é também uma solução ótima do problema de PI.

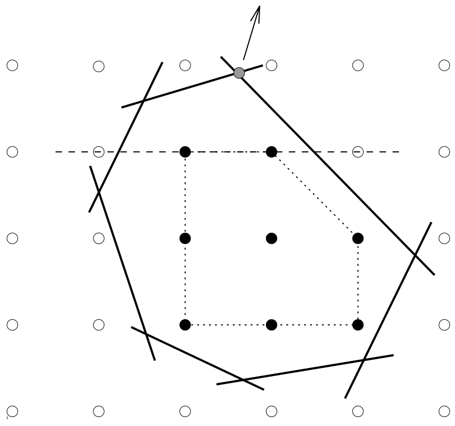
1.3.3 A Equivalência entre Otimização e Separação

Na seção anterior mencionamos que na maioria dos casos o número de facetas de um poliedro é exponencial. Nesta seção apresentamos um resultado fundamental para teoria de combinatória poliédrica, o qual mostra que, dentro da abordagem de planos-de-cortes, a complexidade de um problema de otimização está mais intimamente ligada à complexidade de se efetuar um corte do que ao número de cortes efetuados.

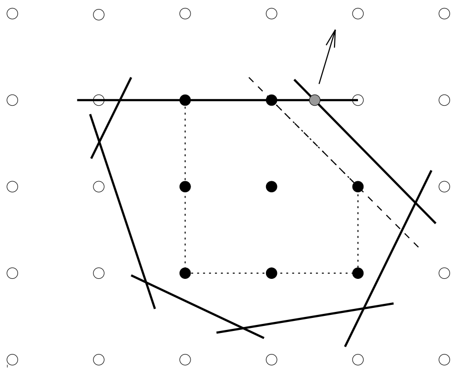
Considere os seguintes problemas:

Problema da Otimização para uma Família de Poliedros:

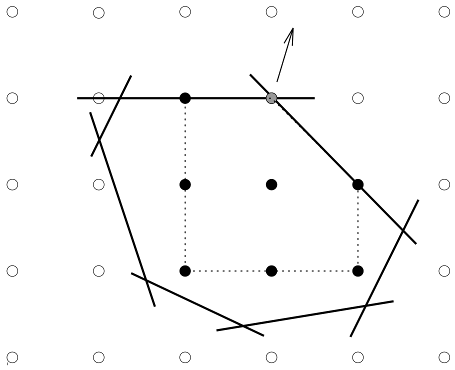
Dado um vetor $c \in I\!R^n$ e um poliedro $Q \subseteq I\!R^n$ da família especificada, determinar o ponto $x^* \in Q$ tal que $cx \leq cx^*$ para todo $x \in Q$.



(a)



(b)



(c)

Figura 1.3: Esboço do funcionamento de um algoritmo *cutting-planes* que utiliza desigualdades que definem facetas.

Problema da Separação para uma Família de Poliedros:
Dado um ponto $y \in \mathbb{R}^n$ e um poliedro $Q \subseteq \mathbb{R}^n$ da família especificada, decidir se y está, ou não, em Q e, em caso negativo, encontrar uma inequação $\pi x \leq \pi_0$ válida com respeito a Q tal que $\pi y > \pi_0$.

O seguinte teorema, devido à Grötschel, Lovász e Schrijver [GLS88], mostra que estes problemas possuem complexidades equivalentes.

Teorema 1.5 *Dada uma família de poliedros, existe um algoritmo polinomial para o problema da otimização se e somente se existe um algoritmo polinomial para o problema da separação.*

Dentre as implicações que advém deste resultado ressaltamos que, para poliedros associados a problemas *NP-difíceis* não devemos esperar que algoritmos polinomiais para o problema da separação sejam encontrados, a menos que $P = NP$. Desta forma, se o problema de PI é *NP-difícil* e estamos utilizando um dado algoritmo *cutting-planes* para resolvê-lo, então deve existir algumas instâncias para as quais este algoritmo terminará com uma solução fracionária. Suponha que $\mathcal{F}$ é o conjunto de famílias de desigualdades que este algoritmo utiliza como corte, para que o algoritmo seja capaz de eliminar todo ponto fracionário obtido no processo sempre deve existir uma desigualdade em $\mathcal{F}$ que elimine estes pontos, assim, mesmo que para algumas das famílias de inequações em $\mathcal{F}$ o problema da separação seja polinomial, pelo Teorema 1.5, devem existir outras famílias para as quais o problema de separação é *NP-difícil.*

1.4 Organização da Dissertação

Os capítulos restantes desta dissertação estão organizados tal como se segue.

No Capítulo 2 apresentamos uma coletânea dos principais resultados e métodos de resolução referentes ao PTCM. Apresentamos versões generalizadas do PTCM que são *NP-difíceis*, discutimos os principais métodos de resolução aproximada e apresentamos os principais avanços no sentido de se resolver este problema à otimalidade.

No Capítulo 3 apresentamos o primeiro dos dois modelos de PI que propomos para o problema, o modelo baseado em segmentos. O nome dado ao modelo advém da associação realizada entre suas variáveis e os segmento em $S(P)$. Além de mostrarmos como formular o problema através de PI, mostramos como fortalecer o modelo através da inserção de desigualdades válidas fortes para o problema. Mostramos ainda que algumas destas desigualdades definem facetas no politopo monótono e apresentamos exemplos do impacto computacional da inserção destas desigualdades no modelo proposto.

No Capítulo 4 apresentamos o modelo baseado em triângulos, modelo no qual as variáveis estão associadas aos triângulos em $\triangle(P)$. Inicialmente, mostramos que em decorrência da semelhança entre a definição do problema sobre triângulos e sobre segmentos, os principais resultados (desigualdades) válidos para o modelo baseado em segmentos, quando devidamente reinterpretados, também são válidos para o modelo baseado em triângulos. Em seguida, baseados no trabalho de Loera et. al [dLHSS96], apresentamos outras possibilidades de formulações de PI oferecidas por este modelo. Como veremos no capítulo que trata dos resultados computacionais, foi justamente com uma destas possibilidades que obtivemos nossos melhores resultados. Finalmente, concluímos o capítulo apresentando uma classe de desigualdades que não é válida para o politopo das triangulações planares, como todas as outras apresentadas, mas sim para o politopo das *triangulações minimamente locais*, uma classe especial de triangulações planares da qual a TCM faz parte.

No Capítulo 5 apresentamos os resultados computacionais obtidos utilizando os modelos propostos e descrevemos os principais aspectos dos algoritmos empregados nos experimentos.

Finalmente, no Capítulo 6 apresentamos nossas conclusões e discorremos sobre possíveis extensões deste trabalho.

Capítulo 2

Abordagens Anteriores

Apresentamos neste capítulo uma compilação dos principais trabalhos referentes ao PTCM encontrados na literatura. Nosso intuito é fornecer ao leitor uma visão geral do que se conhece sobre o problema. Para isto, dividimos a compilação em três partes. Na Seção 2.1 apresentamos problemas correlatos ao PTCM os quais são comprovadamente *NP-difíceis*. A Seção 2.2 é dedicada à apresentação dos principais métodos heurísticos utilizados na resolução do problema e, finalmente, na Seção 2.3 discutimos os principais resultados relativos à resolução exata do PTCM.

2.1 Problemas Correlatos NP-difíceis

A prova de que um certo problema pertence à classe dos problemas *NP-difíceis* invariavelmente exige o estudo de outros problemas a este associados os quais estão nesta classe. Realizar tal prova consiste em mostrar ser possível reduzir polinomialmente o problema *NP-difícil* ao problema considerado. Ou seja, mostrar ser possível, em tempo polinomial, transformar uma instância qualquer do problema *NP-difícil* em uma instância do problema considerado e com base na resposta do último determinar a resposta do primeiro[1]. Assim, o fato de um certo problema cuja complexidade é desconhecida ser semelhante, em termos da instância fornecida e da resposta exigida, a um problema *NP-difícil* pode ser um indício de que o primeiro problema também o seja.

Com base nesta observação e no sentido de mostrar ser o PTCM um problema *NP-difícil*, alguns autores dedicaram-se à identificação de problemas envolvendo triangulações

[1] Para maiores informações a respeito do estudo da complexidade de problemas e provas de *NP-completude* sugerimos [HS84].

planares os quais fossem comprovadamente *NP-difíceis*. Apresentamos na seqüência os três principais problemas derivados deste esforço.

Problema da Existência de uma Triangulação (PET)
Instância: Um conjunto finito P de pontos no plano e um subconjunto de segmentos de reta $S' \subseteq S(P)$.
Questão: O subconjunto S' contém alguma triangulação planar de P?

Este problema foi introduzido originalmente por Lloyd em [Llo77]. Neste trabalho o autor mostra, através de uma redução do *problema da satisfatibilidade* (SAT), que o PET é um problema *NP-completo*.

Apesar de envolver triangulações planares, o PET não se assemelha muito ao PTCM, no sentido em que na instância não são considerados todos os segmentos em $S(P)$ e na resposta não se discute o custo da triangulação, e sim sua existência. Contudo, este resultado apresenta seu maior mérito no fato de possibilitar a prova de que certas generalizações do PTCM são *NP-difíceis*.

Na primeira destas generalizações consideramos todos os segmentos em $S(P)$, porém permitimos que os custos atribuídos a estes segmentos sejam arbitrários.

Problema da Triangulação de custo mínimo com Custos Arbitrários (PTCA)[2]
Instância: Um conjunto finito P de pontos no plano, uma função custo $f : S(P) \to \mathbb{Z}_+$ e um inteiro positivo k.
Questão: Existe uma triangulação planar $T \subseteq S(P)$ de P cujo custo total $\sum_{s \in T} f(s)$ seja menor ou igual a k?

Teorema 2.1 *O PTCA é* NP-completo.

Prova: Como todo segmento em $S(P)$ possui custo inteiro, o custo total de T pode facilmente ser computado em tempo polinomial. Assim, o problema claramente está em *NP*. Para completar a prova mostramos ser possível reduzir polinomialmente o PET ao PTCA.

Seja (P, S') uma instância do PET. Definimos a função de custo f da seguinte forma:

$$f(s) = \begin{cases} 0, & \text{se } s \in S', \\ 1, & \text{caso contrário.} \end{cases}$$

[2] O conceito de *triangulação de custo mínimo com custos arbitrários* não é encontrado na literatura. A apresentação do PTCA neste capítulo se deve ao fato de que a prova de sua *NP-completude* é essencialmente idêntica à prova do Teorema 6.1 apresentado por Heath e Pemmaraju em [HP94].

Considere agora a seguinte instância para o PTCA: $(P, f, k = 0)$. É fácil observar que existe uma triangulação planar de P em S' se e somente se a resposta do PTCA for afirmativa, i.e., existe uma triangulação planar $T \subseteq S(P)$ tal que $\sum_{s \in T} f(s) \leq 0$. Conseqüentemente, o PET pode ser reduzido polinomialmente ao PTCA, o que conclui a prova. □

O principal problema desta generalização, para o propósito de mostrar-se que o PTCM é um problema *NP-difícil*, é o fato de não considerar uma propriedade inerentemente geométrica do PTCM: neste último o custo de um segmento é determinado pela distância euclideana de seus extremos. Informações geométricas como esta podem ser decisivas na complexidade de um certo problema.

Uma outra generalização para o PTCM que contorna este empecilho foi proposta por Heath e Pemmaraju [HP94]. Nesta generalização não são considerados todos os segmentos em $S(P)$, porém os custos utilizados são os mesmos do PTCM.

Problema da Triangulação de Custo Mínimo Generalizado (G-PTCM)
Instância: Um conjunto finito P de pontos no plano, um subconjunto de segmentos $S' \subseteq S(P)$, o qual contém pelo menos uma triangulação de P, e um racional positivo k.
Questão: Existe uma triangulação em S' cujo custo, dado pela soma total dos comprimentos de seus segmentos de reta, não exceda a k?

Através de uma redução do PET os autores mostram que o G-PTCM é *NP-difícil*. A proximidade entre o PTCM e o G-PTCM levou estes mesmos autores a conjecturar que também o PTCM é um problema *NP-difícil*.

2.2 Resolução Aproximada

Em face à inexistência de algoritmos eficientes (polinomiais) capazes de resolver o PTCM à otimalidade, uma alternativa natural é tentar resolver o problema de forma aproximada. Nesta abordagem, o principal objetivo é identificar métodos eficientes capazes de produzir soluções que se aproximem da solução ótima. No caso do PTCM, estes métodos devem produzir triangulações planares cujos custos se aproximem do custo da TCM. As mais tradicionais aproximações utilizadas com este intuito encontradas na literatura são as triangulações de Delaunay e gulosa. Nesta seção descrevemos estas duas aproximações, como computá-las e a garantia de qualidade que oferecem. Além disso, discutimos uma variação da triangulação gulosa, a qual dentre as aproximações conhecidas para o PTCM é a que oferece a melhor garantia de qualidade.

2.2.1 A Triangulação de Delaunay

Tradicionalmente a *triangulação de Delaunay* é definida como o dual de um outro objeto clássico em geometria computacional: o diagrama de Voronoi. Seja P um conjunto finito de pontos no plano. O diagrama de Voronoi de P é uma coleção de regiões R_p do plano, uma para cada ponto $p \in P$, tal que um ponto qualquer q do plano está na região R_p se e somente se o ponto p é o ponto mais próximo a q dentre todos os pontos de P.

A triangulação de Delaunay é então definida da seguinte forma. Dado o diagrama de Voronoi V de P, dois pontos $t, u \in P$ estão ligados por um segmento na triangulação de Delaunay se e somente se as regiões R_t e R_u são vizinhas em V. Na Figura 2.1 encontramos uma ilustração gráfica do enunciado acima, onde as linhas tracejadas representam o diagrama de Voronoi, enquanto as linhas cheias representam os segmentos da triangulação de Delaunay.

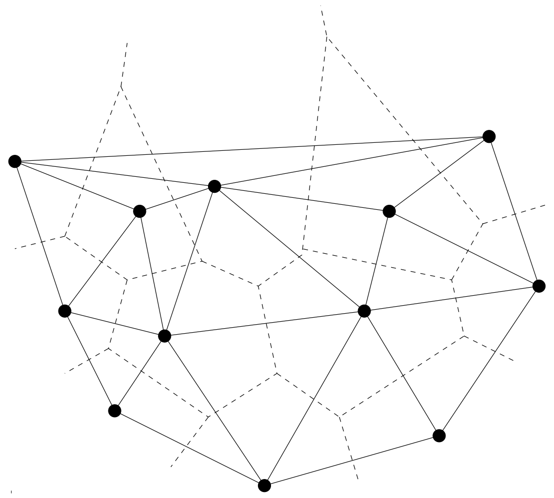


Figura 2.1: Diagrama de Voronoi e triangulação de Delaunay.

Alternativamente, a triangulação de Delaunay pode ser definida como a triangulação planar D, onde $\overline{pq} \in D$ se e somente se existe um círculo que passa por p e q que não contém nenhum outro ponto de P. Utilizando esta última definição podemos computar a triangulação de Delaunay através de um algoritmo baseado na estratégia de divisão e conquista [dRS94].

Inicialmente, particionamos o conjunto P em dois subconjuntos P' e P'' com aproximadamente a mesma cardinalidade, através de uma reta r no plano. Então, construímos de forma recursiva as triangulações de Delaunay de P' e P'', respectivamente D' e D'', e as utilizamos para montar a triangulação de Delaunay de P.

Na realização deste último passo utilizamos o conceito de bolha ambulante. De posse de D' e D'', utilizamos um círculo que percorre o espaço entre as duas triangulações, mantendo sempre contato com pelo menos um vértice de cada uma, sem contudo incluir outros vértices em seu interior. Os pares de vértices tocados pelo círculo determinam os segmentos a serem acrescentados entre D' e D'' a fim de formar a triangulação de Delaunay de P. Neste processo, os segmentos de D' e D'' que interceptam os novos segmentos inseridos devem ser descartados.

O algoritmo acima foi proposto originalmente por Guibas e Stolfi em [GS85]. Neste trabalho os autores mostram que este algoritmo pode ser implementado de forma a computar a triangulação de Delaunay em tempo $O(n \lg n)$ utilizando $O(n)$ em armazenamento. Em termos de complexidade, este algoritmo se equivale aos melhores existentes.

Qualidade da Solução Obtida

Através do contra-exemplo exibido na Figura 2.2, Lloyd [Llo77] mostrou não ser a triangulação de Delaunay equivalente à TCM. Neste exemplo, é fácil observar que a triangulação (a) tem custo menor que a triangulação de Delaunay vista em (b) (onde as linhas tracejadas representam o diagrama de Voronoi).

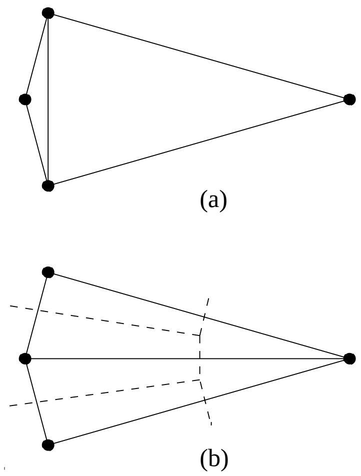


Figura 2.2: Contra-exemplo da triangulação de Delaunay.

Para instâncias arbitrárias, Manacher e Zobrist [MZ79] mostram que o custo da triangulação de Delaunay pode ser pelo menos um fator de $\Omega(n/\lg n)$ maior que a TCM. Posteriormente, Kirkpatrick [Kir80] fortaleceu este resultado mostrando que a triangulação de Delaunay pode ser pelo menos um fator de $\Omega(n)$ maior que a TCM. De fato, até o momento não existe prova de que a triangulação de Delaunay ofereça qualquer garantia de qualidade.

2.2.2 A Triangulação Gulosa

A *triangulação gulosa*, como o nome sugere, é uma triangulação planar derivada da aplicação direta da estratégia gulosa. Nesta estratégia o conjunto solução é construído de forma incremental onde, iniciando-se com um conjunto vazio, a cada passo o algoritmo utiliza-se de um critério de otimização local para conduzir de uma solução parcial à outra. No caso da triangulação gulosa, começando-se com um conjunto vazio, a solução é construída através de um algoritmo que a cada passo adiciona ao conjunto o menor segmento que não intercepta nenhum dos segmentos previamente adicionados.

De forma direta, a triangulação gulosa pode ser computada como se segue. Inicialmente, ordena-se os $\binom{n}{2}$ segmentos em ordem não decrescente de custos. Em seguida, varre-se estes segmentos nesta ordem. Para cada segmento selecionado testa-se sua compatibilidade com os segmentos previamente adicionados à triangulação, e caso não se interceptem adiciona-se o segmento ao conjunto. Não é difícil verificar que este algoritmo requer espaço $O(n^2)$ (número de segmentos ligando os pontos considerados) e tempo

$$t(n) = O(n^2 \lg n + n^2 f(n) + ng(n)),$$

onde $n^2 \lg n$ é tempo gasto para ordenar os segmentos, $f(n)$ é o tempo gasto para testar a compatibilidade de cada segmento selecionado com os previamente adicionados, e $g(n)$ é o tempo necessário para atualizar a estrutura de dados utilizada no processo, sempre que um novo segmento é adicionado à solução parcial.

Ingenuamente, podemos implementar o algoritmo acima da seguinte forma. Armazenando-se os segmentos que compõem a triangulação em uma lista (tempo necessário para atualização $O(1)$), a cada segmento selecionado no processo de varredura testa-se se este segmento intercepta algum dos segmentos contidos na lista (no máximo $O(n)$ segmentos). Claramente, esta implementação possui uma complexidade de tempo $O(n^3)$. Contudo, Gilbert [Gil79] propõe uma estrutura de dados mais refinada, através da qual é possível realizar o teste de compatibilidade em $O(\lg n)$ gastando-se somente $O(n \lg n)$ na atualização, baixando-se assim o tempo requerido pelo algoritmo para $O(n^2 \lg n)$.

Uma abordagem alternativa a gerar todos os segmentos e então de forma iterativa determinar aqueles que compõem a triangulação gulosa é gerar apenas segmentos compatíveis, i.e., segmentos que com certeza estão em uma mesma triangulação gulosa. De fato, utilizando esta abordagem, Levcopoulos e Krznaric [LK94] mostraram que a triangulação gulosa pode ser computada a partir da triangulação de Delaunay em tempo linear, o que implica que também a triangulação gulosa pode ser computada em $O(n \lg n)$ utilizando apenas $O(n)$ em armazenamento.

Qualidade da Solução Obtida

Assim como a triangulação de Delaunay, a triangulação gulosa também não equivale à TCM, como pode ser observado no contra-exemplo da Figura 2.3 [Llo77] (onde são dadas as coordenadas de cada ponto). No exemplo, a TCM é dada pelos segmentos { $\overline{ab}$, $\overline{bc}$, $\overline{cd}$, $\overline{de}$, $\overline{ae}$, $\overline{ac}$, $\overline{ce}$ }, e a triangulação gulosa pelos segmentos { $\overline{ab}$, $\overline{bc}$, $\overline{cd}$, $\overline{de}$, $\overline{ae}$, $\overline{bd}$, $\overline{be}$ }. As duas triangulações diferem apenas nos segmentos interiores. No entanto, enquanto o custo total dos segmentos $\overline{ac}$ e $\overline{ce}$ é aproximadamente 172 o custo total de $\overline{bd}$ e $\overline{be}$ é aproximadamente 187.

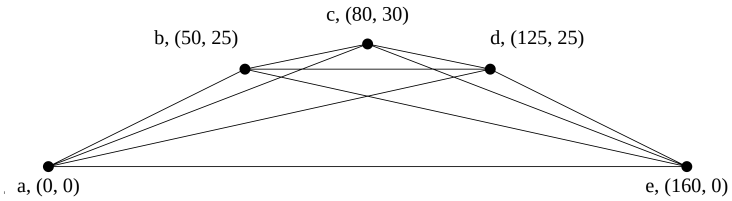


Figura 2.3: Contra-exemplo da triangulação gulosa.

Em [MZ79], Manacher e Zobrist mostram que existe uma configuração de pontos para a qual o custo da triangulação gulosa pode ser pelo menos um fator de $\Omega(n^{1/3})$ maior que a TCM. Fortalecendo este resultado, Levcopoulos [Lev87] mostrou que este limite pode ser de até $\Omega(\sqrt{n})$. Recentemente, Levcopoulos e Krznaric [LK96a] mostraram que este último limite é o pior caso da triangulação gulosa, i.e., a razão entre o custo de uma triangulação gulosa e de uma TCM não é maior que $O(\sqrt{n})$.

2.2.3 A Triangulação Quase-Gulosa

Apesar das triangulações gulosa e de Delaunay não oferecerem boas garantias de qualidade, resultados empíricos parecem indicar que elas funcionam bem na prática. De fato, se os pontos estão uniformemente distribuídos, o custo esperado destas duas aproximações é um fator constante do custo da TCM [LL91]. Além disso, se os pontos de P estão sobre a fronteira de sua envoltória o custo da triangulação gulosa não excede a um fator constante do custo da TCM [LL87].

Com relação à triangulação gulosa, confirmando as expectativas fomentadas pelas observações acima, Levcopoulos e Krznaric [LK96a] mostraram que a configuração de pontos apresentada em [Lev87] é a única na qual a razão entre o custo da triangulação gulosa e o da TCM não é limitado por $O(1)$. Baseados nesta constatação os autores propõem uma alteração nos algoritmos utilizados para computar a triangulação gulosa, de forma a evitar este caso patológico. A triangulação produzida por este algoritmo é chamada pelos autores de *triangulação quase-gulosa*.

2.3 Resolução Exata

O fato de não se conhecer algoritmo polinomial capaz de computar a TCM de um conjunto arbitrário de pontos, não se deve à inexistência de trabalhos que abordem o tema. É fato que o número de trabalhos na literatura que abordam a resolução aproximada do problema é significativamente maior que o número de trabalhos que abordam a resolução exata. Contudo, trabalhos desta segunda linha têm apresentado relevantes resultados, os quais, ocasionalmente, têm sido extremamente úteis mesmo no aprimoramento dos métodos aproximados existentes [HP94]. Nesta seção sumarizamos estes resultados.

2.3.1 Casos Polinomiais

O primeiro resultado importante quanto à resolução exata do PTCM foi apresentado em [Gil79]. Neste trabalho Gilbert mostrou ser possível, através de uma abordagem de programação dinâmica, computar a TCM de um polígono simples em tempo polinomial.

Seja P um polígono simples com vértices $p_0, p_1, ..., p_{n-1}$. O algoritmo proposto baseia-se no fato de que o problema sobre P pode ser definido recursivamente sobre subpolígonos simples de P (polígonos cujos segmentos são lados ou diagonais de P). Seja $C[i, i+j]$ o custo da TCM de um polígono P' limitado por $\overline{p_{i+j}p_i}$ e pelos segmentos de $\overline{p_ip_{i+1}}$ até $\overline{p_{i+j-1}p_{i+j}}$, onde os índices são módulo n. Se $\overline{p_ip_{i+j}}$ intercepta algum dos lados de P, então P' não é um subpolígono de P, e assim fazemos $C[i, i+j] := \infty$. Caso contrário, $\overline{p_ip_{i+j}}$ é um lado ou uma diagonal de P. Em ambos os casos, $\overline{p_ip_{i+j}}$ é um dos lados de algum triângulo na TCM de P', e este triângulo particiona P' em dois subpolígonos menores (possivelmente vazios), cujas triangulações devem ser ótimas. Assim temos que:

$$C[i, i+j] := |\ \overline{p_{i+j}p_i}\ | + \min_{k=1..j-1}(C[i, i+k] + C[i+k, i+j]).$$

Na Figura 2.4 apresentamos o algoritmo que utiliza os resultados acima para computar a TCM de P. Neste algoritmo a TCM é armazenada em $T[0, n-1]$ e seu custo em $C[0, n-1]$. Devido à necessidade de armazenar a tabela $C[i, j]$ o algoritmo requer espaço $O(n^2)$. A complexidade de tempo do algoritmo é limitada pelo passo 2 o qual, devido à existência de 3 laços aninhados, sobre as variáveis i, j e k (operação *min*), requer tempo $O(n^3)$.

1. Para $i := 0$ até $n-1$ faça

   $C[i,i] := 0.$

2. Para $j := 1$ até $n-1$ faça

   Para $i := 1$ até $n-1$ faça

   $$c(\ \overline{p_{i+j}p_i}\ ) := \begin{cases} |\ \overline{p_{i+j}p_i}\ |, & \text{se } \overline{p_{i+j}p_i} \text{ está totalmente contido em } P\ , \\ \infty, & \text{caso contrário.} \end{cases}$$

   $C[i,i+j] := c(\ \overline{p_{i+j}p_i}\ ) + \min_{k=1..j-1}(C[i,i+k] + C[i+k,i+j]).$

   $T[i,i+j] := \{\ \overline{p_{i+j}p_i}\ \} \cup T[i,i+k] \cup T[i+k,i+j].$

Figura 2.4: Algoritmo de programação dinâmica para determinar a TCM de um polígono simples.

O algoritmo proposto por Gilbert foi posteriormente alterado de forma a possibilitar a computação de um objeto mais genérico que um polígono simples, uma célula [HP94]. Uma *célula* de um conjunto de pontos P consiste em um conjunto de segmentos que formam uma árvore geradora do grafo $G = (P, S(P))$ acrescido dos segmentos que limitam a envoltória convexa de P. Toda célula pode ser unicamente representada através de uma seqüência de vértices $p_0, p_1, ..., p_{n-1}$ obtida percorrendo-se esta célula em algum sentido, p.ex., o anti-horário. Para exemplificação, considere a célula exibida na Figura 2.5. Esta célula pode ser representada através da seqüência 1, 2, 3, 2, 1, 4, 5, 4, 6, 7, 8, 9, 10, 9, 8, 7, 11, 12, 13, 14, 13, 12, 11, 15. Note que esta seqüência não é composta exclusivamente de vértices distintos. Contudo, para os propósitos do algoritmo cada elemento da seqüência é tratado como um vértice distinto. De posse da seqüência que define uma célula, o algoritmo de programação dinâmica que computa o restante dos segmentos da TCM é similar ao apresentado na Figura 2.4, exceto que agora os segmentos em $S(P)$ que interceptam algum dos segmentos que compõem a célula não devem ser considerados, o que pode ser feito atribuindo-se um custo suficientemente elevado a estes segmentos.

Com base no resultado acima, vemos que computar a TCM em tempo polinomial equivale a determinar um conjunto $T' \subseteq S(P)$ de segmentos que está contido em uma TCM, tal que o grafo $G = (P, T')$ seja conexo, pois neste caso T' contém uma célula e o restante dos segmentos da TCM podem ser computados em $O(n^3)$. Fortalecendo este resultado, Cheng, Golin e Tsang [CGT95] mostraram que mesmo que G não seja conexo o problema pode ser resolvido em $O(n^{k+2})$, onde k é o número de componentes conexos de G.

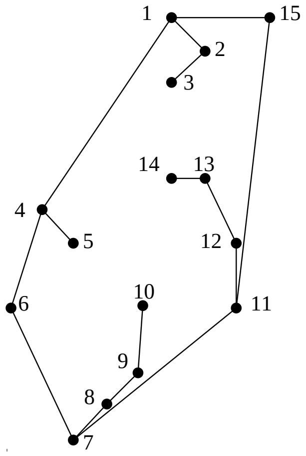


Figura 2.5: Exemplo de uma célula de 15 pontos.

2.3.2 Determinando Subconjuntos não Triviais da TCM

A possibilidade de computar a TCM em tempo polinomial a partir de um subconjunto de seus segmentos motivou trabalhos no sentido de determinar conjuntos de segmentos $T' \subseteq S(P)$ que estão em alguma TCM. Os segmentos que limitam a envoltória convexa de P formam um subconjunto trivial de qualquer TCM. Gilbert [Gil79] mostrou que o menor segmento em $S(P)$ também está em qualquer TCM. No restante desta seção descrevemos dois métodos capazes de computar subconjuntos da TCM menos triviais que estes dois.

O β-Esqueleto

O conceito de β-esqueleto foi proposto originalmente por Kirkpatrick e Radke [KR85]. Sejam $p, q \in P$. A *vizinhança proibida* $F(p,q)$ de p e q é definida como a união de dois discos de raio $\beta|\overline{pq}|/2$, com $\beta \geq 1$, que passam sobre p e q (Figura 2.6). O β-esqueleto de P é definido como sendo o seguinte conjunto de segmentos:

$$B(P) = \{\ \overline{pq}\ \in S(P) : F(p,q) \cap (P \setminus \{p,q\}) = \emptyset\}.$$

Em [Kei94], Keil mostra que o $\sqrt{2}$-esqueleto está contido em qualquer TCM. Mais recentemente, Cheng e Xu [CX96] melhoraram este resultado provando que o β-esqueleto está contido em qualquer TCM quando $\beta \geq 1.17682$.

Note que, por definição, se um certo segmento $\overline{pq}$ está em um β-esqueleto, então existem dois círculos que passam por p e q e que não contêm nenhum outro ponto de P em seu interior. Ocorre, que a existência de um destes círculos já é condição suficiente

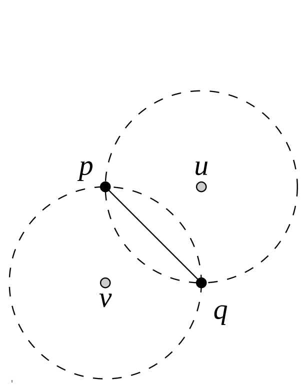


Figura 2.6: A vizinhança proibida $F(p,q)$, $|\overline{pu}| = |\overline{pv}| = \beta|\overline{pq}|/2$.

para que o segmento $\overline{pq}$ pertença a triangulação de Delaunay de P. Em outras palavras, um β-esqueleto sempre é um subconjunto da triangulação de Delaunay. Kirkpatrick e Radke [KR85] mostraram que este fato possibilita computar tais esqueletos a partir da triangulação de Delaunay em tempo linear, ou seja, em tempo total $O(n \lg n)$.

Na Figura 2.9(a) apresentamos o 1.17682-esqueleto de um conjunto de 80 pontos. Como pode se observar este conjunto de segmentos não conecta todos os pontos considerados, mesmo que a ele adicionemos os segmentos que limitam a envoltória convexa. De fato, Cheng, Golin e Tsang [CGT95] mostraram que no caso onde os pontos estão uniformemente distribuídos o número esperado de componentes conexos no grafo $G = (P, B(P))$, para $\beta \geq 1.17682$, é $\Omega(n)$.

O LMT-Esqueleto

Seja T uma triangulação planar de P. Se o segmento $\overline{pq} \in T$ não está na envoltória convexa de P, então este segmento é um dos lados de dois triângulos vazios formados por T. Juntando estes dois triângulos formamos um quadrilátero, do qual $\overline{pq}$ é uma das diagonais. Caso este quadrilátero seja convexo existe uma outra diagonal $\overline{uv}$. Dizemos que $\overline{pq}$ é minimamente local se o quadrilátero não é convexo (Figura 2.7(a)), ou se, em caso contrário, $|\overline{pq}| \leq |\overline{uv}|$ (Figura 2.7(b)).

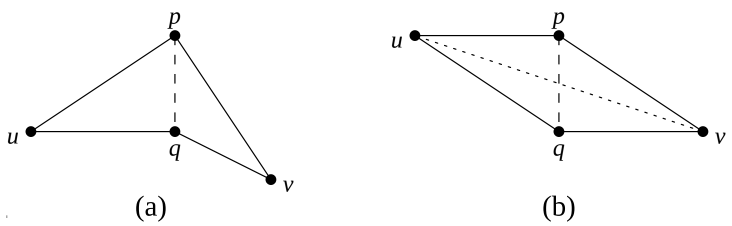


Figura 2.7: Exemplos de segmentos minimamente locais.

Uma triangulação planar onde todos os segmentos são minimamente locais é chamada de *triangulação minimamente local*. Claramente, toda TCM é uma triangulação minimamente local. Infelizmente, a recíproca não é verdadeira, p.ex., a triangulação gulosa é minimamente local, porém, não equivale obrigatoriamente à TCM.

Nos exemplos da Figura 2.7 os quadriláteros apresentados fornecem uma garantia de que o segmento $\overline{pq}$ é minimamente local em alguma triangulação planar de P. Neste caso, dizemos que os triângulos que compõem estes quadriláteros fornecem um certificado para o segmento $\overline{pq}$. É fácil constatar que segmentos que não possuem certificados não podem estar em nenhuma triangulação minimamente local.

Baseado neste fato, Dickerson e Montague [DM96] propõem um algoritmo capaz de identificar segmentos que estão contidos em qualquer triangulação minimamente local, e por conseguinte, na TCM. Este conjunto de segmentos foi denominado pelos autores de LMT-esqueleto. O LMT-esqueleto não corresponde obrigatoriamente ao maior conjunto de segmentos que está contido em qualquer triangulação minimamente local, os segmentos que o compõem são definidos algoritmicamente.

Na Figura 2.8 apresentamos este algoritmo. Devido à existência da lista *candTris*, este algoritmo requer espaço $O(n^3)$. Os passos 1, 2, 3 e 4 podem ser efetuados, respectivamente, em tempo $O(n^3)$ [EORW92], $O(n^2)$, $O(1)$ e $O(n)$. O passo 5 consiste de um laço de $O(n^2)$ iterações: uma para cada segmento s. A complexidade do passo 5(a) é dada pelo número de triângulos que s possui de cada lado, o qual é limitado por $O(n^2)$, enquanto, a complexidade do passo 5(b) é limitado pelo tamanho de *candSegs*, i.e., $O(n^2)$. Como o passo 6 pode ser executado até $O(n^2)$ vezes, o tempo total requerido pelo algoritmo é $O(n^6)$.

A Figura 2.9(b) apresenta um exemplo de um LMT-esqueleto para um conjunto de 80 pontos. Note que este conjunto conecta todos os pontos considerados, de forma, que o restante dos segmentos da TCM podem ser computados em tempo polinomial. De fato, nos testes realizados em [DM96], via de regra, o grafo euclideano $G = (P, T')$, onde T' é o LMT-esqueleto de P, mostrou-se conexo. Contudo, Bose, Devroye e Evans [BDE96] mostraram que para certa configuração específica de pontos o número de componentes conexos deste grafo pode ser $\Omega(n)$.

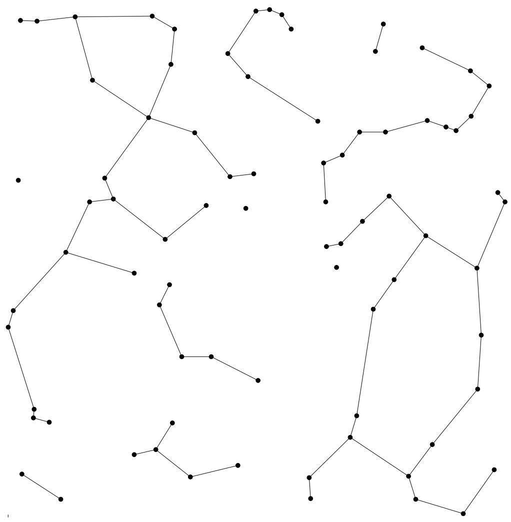
2.3.3 A Abordagem de PI

Pouco se investigou sobre a possibilidade de resolução exata do PTCM através de técnicas de PI. Até o momento da redação deste texto apenas dois trabalhos nesta linha haviam sido identificados.

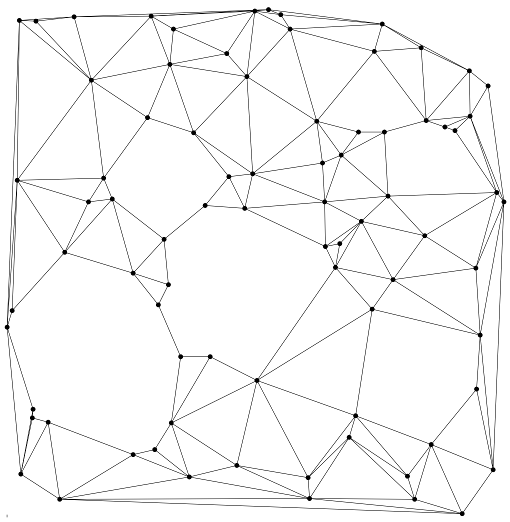
O primeiro destes trabalhos, desenvolvido por Kyoda [Kyo96], é extremamente seme-

1. *candTris* := Uma lista de todos os triângulos vazios com extremos em P.
2. *candSegs* := Uma lista de todos os segmentos de reta com extremos em P.
3. LMT := Uma lista vazia.
4. Remova os segmentos que limitam a envoltória convexa de *candSegs* e insira-os em LMT.
5. Para cada segmento $s \in$ *candSegs* faça
   (a) Se não existem dois triângulos em *candTris* capazes de fornecer um certificado ao segmento s, então remova s de *candSegs* e remova todos os triângulos contendo s de *candTris*.
   (b) Se s não intercepta qualquer outro segmento em *candSegs* ou LMT, então adicione-o a LMT.
6. Se a lista *candSegs* foi alterada no passo 5, então execute o passo 5 novamente.

Figura 2.8: Algoritmo que define o LMT-esqueleto.



(a) 1.17682-esqueleto.



(b) LMT-esqueleto.

Figura 2.9: Exemplos de subconjuntos da TCM de um conjunto de 80 pontos.

lhante ao que aqui realizamos com o modelo baseado em segmentos. Apesar de ambos os trabalhos terem sido desenvolvidos de forma independente, a formulação de PI utilizada por Kyoda é idêntica a apresentada no Capítulo 3 e o algoritmo empregado para resolução computacional também é um *branch-and-cut*. A maior diferença identificada entre os dois trabalhos está nas rotinas de separação desenvolvidas. Contudo, em virtude da pequena quantidade de informações apresentada por Kyoda a cerca do desempenho computacional de sua implementação do *branch-and-cut*, fica difícil realizarmos uma comparação entre os dois trabalhos. Em uma das poucas referências ao desempenho de seu algoritmo, Kyoda diz que foi capaz de resolver o PTCM para instâncias de até 100 pontos. Como veremos no Capítulo 5, utilizando nossa implementação do *branch-and-cut*, fomos capazes de resolver o PTCM para instâncias de até 160 pontos com um tempo médio de computação pouco superior a 2 horas.

O outro trabalho identificado foi realizado por Loera, Hosten, Santos e Sturmfels [dLHSS96]. Neste trabalho os autores também propõem uma formulação de PI que pode modelar o PTCM, mas que não foi originalmente proposta com esta intenção específica. Esta formulação modela uma generalização do conceito de triangulações planares, onde os pontos pertencem a $\mathbb{R}^d$ com $d \geq 2$ e as regiões limitadas formadas (triângulos quando $d = 2$) não são necessariamente vazias. Como já mencionamos, o modelo apresentado no Capítulo 4 é baseado fudamentalmente neste trabalho.

Capítulo 3

O Modelo Baseado em Segmentos

Dedicamos este capítulo ao estudo de um modelo de PI para o PTCM através do qual seja possível identificar o conjunto de segmentos que caracteriza uma TCM. Para simplificar, chamamos este modelo de *modelo baseado em segmentos*.

Inicialmente, na Seção 3.1, mostramos como formular o PTCM como um problema de PI 0-1. Nas seções seguintes mostramos como fortalecer esta formulação inicial através da inserção de outras desigualdades válidas fortes. Mostramos ainda que estas desigualdades, ou casos particulares delas, definem facetas no politopo monótono, e apresentamos exemplos do impacto computacional causado pela inserção de algumas destas desigualdades no modelo de PI proposto.

3.1 Formulando o PTCM através de PI

Considere um conjunto finito ordenado S e um de seus subconjuntos $S' \subseteq S$. Dizemos que o vetor $x^{S'} \in I\!R^{|S|}$ é o *vetor de incidência* de S' se existe uma correspondência biunívoca entre seus componentes e os elementos de S, de tal forma que para cada um destes componentes $x_s^{S'}$ temos que:

$$x_s^{S'} = \begin{cases} 1, & \text{se } s \in S', \\ 0, & \text{se } s \in S \setminus S'. \end{cases}$$

Assim, dada uma triangulação planar $T \subseteq S(P)$ dizemos que $x^T \in I\!R^{|S(P)|}$ é o seu vetor de incidência se:

$$x_{ij}^T = \begin{cases} 1, & \text{se } \overline{ij} \in T, \\ 0, & \text{se } \overline{ij} \in S(P) \backslash T. \end{cases}$$

Seja $\mathcal{X}_{TP}^s$ o conjunto de todos os vetores de incidência de triangulações planares de P. Representando por c_{ij} o custo, ou comprimento, de cada segmento $\overline{ij} \in S(P)$, observamos que determinar o vetor de incidência de uma TCM, e por conseguinte a própria TCM, consiste em resolver o seguinte problema:

$$\begin{array}{ll} \min & \sum_{\overline{ij} \in S(P)} c_{ij} x_{ij} \\ \text{Sujeito a} & x \in \mathcal{X}_{TP}^s. \end{array}$$

Logo, uma maneira de formular o PTCM como um problema de PI consiste em identificar um sistema de desigualdades lineares, cujas variáveis sejam os elementos do vetor x, que juntamente com um conjunto de restrições de integralidade sejam capazes de garantir a restrição $x \in \mathcal{X}_{TP}^s$. Em outras palavras, as propriedades expressas por este sistema de desigualdades devem caracterizar as triangulações planares.

Para que possamos apresentar uma formulação de PI cujas restrições satisfaçam este requisito consideremos a seguinte proposição.

Proposição 3.1 *Seja P um conjunto finito de $n \geq 3$ pontos no plano. Toda triangulação planar de P possui $3(n-1)-h$ segmentos, onde h é o número de segmentos que limitam a envoltória convexa de P.*

Prova: Seja $T \subseteq S(P)$ uma triangulação planar arbitrária de P. Como o grafo euclideano $G = (P, T)$ é planar, se f representa o número de faces de G, pela fórmula de Euler, temos que: $n - |T| + f = 2$. Além disso, é fácil constatar que a soma dos segmentos que limitam todas as faces (3 para cada face limitada e h para a não limitada) é igual a duas vezes o número total de segmentos, i.e., $3(f-1) + h = 2|T|$. De onde segue que $|T| = 3(n-1) - h$. □

Este resultado nos fornece uma forma alternativa de garantir que um conjunto de segmentos $S \subseteq S(P)$ que dois a dois não se interceptam é uma triangulação planar. Representando a cardinalidade de uma triangulação planar de P por $\phi_s(P)$, temos que o PTCM pode ser formulado como o seguinte problema de PI:

$$
\begin{array}{llll}
\min & \displaystyle\sum_{\overline{ij} \in S(P)} c_{ij} x_{ij} & & \\
\text{Sujeito a} & x_{ij} + x_{k\ell} \leq 1, & \forall\, \overline{ij}\ ,\ \overline{k\ell}\ \in S(P),\ \text{tal que}\ \overline{k\ell}\ \cap\ \overline{ij}\ \neq \emptyset, & \text{(I)} \\
& \displaystyle\sum_{\overline{ij} \in S(P)} x_{ij} = \phi_s(P), & & \text{(II)} \\
& x_{ij} \geq 0, & \forall\, \overline{ij}\ \in S(P), & \text{(III)} \\
& x_{ij} \leq 1, & \forall\, \overline{ij}\ \in S(P), & \text{(IV)} \\
& x_{ij} \in \mathbb{Z}, & \forall\, \overline{ij}\ \in S(P). & \text{(V)}
\end{array}
$$

As desigualdades da classe (I) dizem que dois segmentos que se interceptam não podem estar simultaneamente em uma mesma triangulação planar. A igualdade (II), a qual chamamos de *restrição de cardinalidade*, determina o número de segmentos de uma triangulação planar, impondo assim a condição de maximalidade. Finalmente, as restrições das classes (III), (IV) e (V) determinam o conjunto de valores que as variáveis podem assumir, no caso, 0 ou 1.

O seguinte resultado garante a validade desta formulação.

Proposição 3.2 *Dado $S \subseteq S(P)$, x^S satisfaz às restrições da formulação acima se e somente se S é uma triangulação planar de P.*

Prova: ($\Rightarrow$) Se x^S satisfaz às restrições da formulação em questão, então S representa um conjunto de segmentos cuja cardinalidade é $\phi_s(P)$ e tal que quaisquer dois destes segmentos não se interceptam. Pela condição de maximalidade, se S não é uma triangulação planar, então existe uma triangulação planar $T \supset S$ de P cuja cardinalidade é superior à $\phi_s(P)$, fato que contradiz a Proposição 3.1.

($\Leftarrow$) Se S é uma triangulação planar, quaisquer dois de seus segmentos não se interceptam, e assim a desigualdade (I) deve ser satisfeita. Além disso, pela Proposição 3.1, $|S| = \phi_s(P)$, e assim a igualdade (II) deve ser satisfeita. □

Infelizmente, a Proposição 3.2, apesar de garantir a validade, não garante a força da formulação acima. Porém, como mencionamos no Capítulo 1, dentro da abordagem de planos-de-corte, a força do modelo de PI empregado, ou de suas desigualdades, está diretamente relacionada à sua eficiência computacional. Assim, no restante deste capítulo discutimos a força das desigualdades que compõem esta formulação e as possibilidades de fortalecê-la através da inserção de novas desigualdades válidas fortes. Antes, porém, algumas considerações são necessárias.

A primeira destas considerações diz respeito à dimensão do politopo $conv(\mathcal{X}^s_{TP})$, o qual representaremos por Q^s_{TP}. O fato da equação (II) ser válida com respeito a Q^s_{TP} indica que este politopo não possui dimensão plena. Isto, como vimos no Capítulo 1, representa uma dificuldade adicional para a aplicação do método indireto na caracterização de desigualdades que definem facetas. Assim, optamos por validar a força das desigualdades apresentadas sobre o *politopo monótono*, i.e., o politopo definido pela envoltória convexa dos vetores de incidência associados aos subconjuntos de triangulações planares. Seja $\tilde{Q}^s_{TP}$ este politopo, o seguinte resultado encoraja esta opção.

Proposição 3.3 *O politopo $\tilde{Q}^s_{TP}$ possui dimensão plena.*

Prova: Considere os vetores de incidência do conjunto vazio e dos conjuntos $\{\overline{ij}\}$, para todo $\overline{ij} \in S(P)$. Claramente, estes vetores pertencem a $\tilde{Q}^s_{TP}$ e são afim independentes. Logo, $\tilde{Q}^s_{TP}$ possui dimensão plena. □

Além das facilidades técnicas que advém deste resultado, $\tilde{Q}^s_{TP}$ também é de interesse prático, pois dado um vetor de custos $c \in \mathbb{R}^n$ é possível construirmos um outro vetor $\overline{c} \in \mathbb{R}^n$ tal que x^* é solução ótima da função cx em Q^s_{TP} se e somente se é solução ótima da função $\overline{c}x$ em $\tilde{Q}^s_{TP}$.

Proposição 3.4 *Para qualquer $c \in \mathbb{R}^n$ e $d > \max\{c_{ij} : \overline{ij} \in S(P)\}$, as seguintes afirmações são equivalentes:*

(i) x^* é uma solução ótima para $\min\{cx : x \in Q^s_{TP}\}$.

(ii) x^* é uma solução ótima para $\max\{\overline{c}x : x \in Q^s_{TP}\}$, onde $\overline{c}_{ij} = d - c_{ij}$ para todo $\overline{ij} \in S(P)$.

(iii) x^* é uma solução ótima para $\max\{\overline{c}x : x \in \tilde{Q}^s_{TP}\}$.

Prova: (i) $\Leftrightarrow$ (ii). x^* é uma solução ótima para $\min\{cx : x \in Q^s_{TP}\}$ se e somente se é uma solução ótima para $\max\{-cx : x \in Q^s_{TP}\}$. Porém, para qualquer $x \in Q^s_{TP}$ temos que $\sum_{\overline{ij} \in S(P)} dx_{ij} = d\phi_s(P)$, e assim (i) e (ii) são equivalentes.

(ii) $\Leftrightarrow$ (iii). Como $\overline{c}_{ij} > 0$ para todo $\overline{ij} \in S(P)$, se x^* é uma solução ótima para $\max\{\overline{c}x : x \in \tilde{Q}^s_{TP}\}$, então x^* é um elemento maximal de $\tilde{Q}^s_{TP}$. Contudo, $x^* \in Q^s_{TP}$ se e somente se é um elemento maximal de $\tilde{Q}^s_{TP}$. □

Um problema potencial na utilização de $\tilde{Q}^s_{TP}$ para validar a força de desigualdades válidas para Q^s_{TP} reside no fato de que, em alguns casos, tais desigualdades não são válidas para $\tilde{Q}^s_{TP}$. Quando nos deparamos com uma desigualdade deste tipo, procuramos identificar uma outra desigualdade válida para $\tilde{Q}^s_{TP}$ que defina a mesma face em Q^s_{TP} que a desigualdade original. Se tal desigualdade é determinada, como ambas definem a mesma face, podemos utilizar a nova desigualdade para avaliar a força da desigualdade original.

No caso específico das equações lineares válidas para Q^s_{TP}, as quais não são válidas para $\tilde{Q}^s_{TP}$, visto que este politopo possui dimensão plena, a solução para o problema mencionado no parágrafo anterior consiste tão somente em transformar a igualdade em uma desigualdade do tipo '$\leq$'. Assim, a desigualdade válida para $\tilde{Q}^s_{TP}$ que define a mesma face que a equação (II), válida para Q^s_{TP}, é

$$\sum_{\overline{ij} \in S(P)} x_{ij} \leq \phi_s(P). \tag{VI}$$

Como veremos tanto a desigualdade (VI) quanto a desigualdade (I) são casos particulares de classes de desigualdades mais genéricas, as quais serão estudadas nas seções 3.4 e 3.2.2, respectivamente. Assim, nos limitamos aqui a discutir a força das desigualdades (III) e (IV), as quais claramente são válidas tanto para para Q^s_{TP} quanto para $\tilde{Q}^s_{TP}$.

Proposição 3.5 *Para todo segmento* $\overline{ij} \in S(P)$ *a desigualdade (III) define faceta em* $\tilde{Q}^s_{TP}$.

Prova: Note que para todo segmento $\overline{ij} \in S(P)$ os vetores de incidência do conjunto vazio e dos conjuntos $\{\overline{k\ell}\}$, onde $\overline{k\ell} \in S(P) \setminus \{\overline{ij}\}$, são afim independentes, pertencem a $\tilde{Q}^s_{TP}$ e satisfazem $x_{ij} = 0$. Além disso, como o vetor de incidência do conjunto $\{\overline{ij}\}$ também pertence a $\tilde{Q}^s_{TP}$, a face definida pela inequação em questão é própria, e conseqüentemente é uma faceta. □

Proposição 3.6 *Seja* $\overline{ij} \in S(P)$ *e* $I(\overline{ij}) = \{\overline{k\ell} \in S(P) \setminus \overline{ij} : \overline{ij} \cap \overline{k\ell} \neq \emptyset\}$. *A face definida em* $\tilde{Q}^s_{TP}$ *pela desigualdade (IV) referente a* $\overline{ij}$ *possui dimensão igual a* $|S(P) \setminus I(\overline{ij})|$.

Prova: Considere os conjuntos $\{\overline{ij}\}$ e $\{\overline{ij}, \overline{k\ell}\}$, onde $\overline{ij}, \overline{k\ell} \in S(P)$, $\overline{ij} \neq \overline{k\ell}$ e $\overline{k\ell} \notin I(\overline{ij})$. Claramente, os $|S(P) \setminus I(\overline{ij})|$ vetores de incidência destes conjuntos são linearmente independentes, pertencem a $\tilde{Q}^s_{TP}$ e satisfazem $x_{ij} = 1$. Suponha, então, que $\overline{k\ell} \in I(\overline{ij})$. Neste caso, vemos que a face em questão está contida na face definida pela desigualdade $x_{k\ell} \geq 0$. Destes dois casos segue o que queríamos provar. □

Da proposição acima temos que, a desigualdade (IV) referente a $\overline{ij} \in S(P)$ define faceta em $\tilde{Q}^s_{TP}$ se e somente se $I(\overline{ij}) = \emptyset$.

3.2 O Politopo dos Conjuntos Independentes

Por definição, dois segmentos de $S(P)$ estão em uma mesma triangulação planar de P somente se não se interceptam. Mais que isto, dados dois segmentos $\overline{ij}$, $\overline{k\ell} \in S(P)$ que não se interceptam e uma triangulação planar $T \subseteq S(P)$, o fato de que $\overline{ij} \in T$ não é condição suficiente para que $\overline{k\ell} \notin T$. Neste sentido, dados dois segmentos distintos de $S(P)$, se eles não se interceptam dizemos que são independentes.

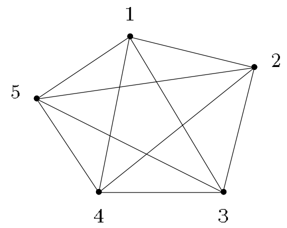
Usando esta nomenclatura podemos dizer que uma triangulação planar de P consiste em um conjunto maximal de segmentos independentes de $S(P)$. De fato, com base nesta observação, podemos associar as triangulações planares de P a conjuntos independentes de um grafo construído a partir da relação de interseção dos segmentos de $S(P)$. Uma implicação interessante, e um tanto imediata, desta associação é que todas as propriedades válidas para conjuntos independentes em grafos arbitrários também são válidas para os conjuntos independentes deste grafo especial, e conseqüentemente são válidas para as triangulações planares. Desta forma, nesta seção visamos apresentar resultados válidos para o politopo dos conjuntos independentes os quais possam ser utilizados para fortalecer o modelo proposto.

3.2.1 Triangulações Planares e Conjuntos Independentes

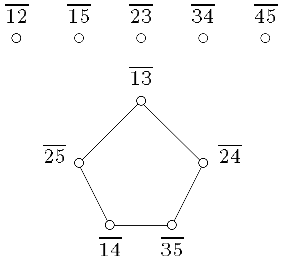
Seja $G = (V, E)$ um grafo simples não-orientado, onde V representa o seu conjunto de vértices e E seu conjunto de arestas. Um *conjunto independente* (*vertex packing, stable set*) de G é um subconjunto $V' \subseteq V$ para o qual $v_i, v_j \in V'$ implica $(v_i, v_j) \notin E$.

Dado um conjunto finito P de $n \geq 3$ pontos no plano, dizemos que $G_P^{is} = (W, H)$ é o *grafo de interseções de segmentos* de P se: para cada segmento $\overline{ij} \in S(P)$ existe um vértice $v_{ij} \in W$ que o representa, e para cada dois segmentos $\overline{ij}$, $\overline{k\ell} \in S(P)$ que se interceptam existe uma aresta $(v_{ij}, v_{k\ell}) \in H$ (Figura 3.1).

Claramente, existe uma bijeção entre as triangulações planares de P e os conjuntos independentes maximais de G_P^{is}. Assim, se atribuirmos a cada vértice $v_{ij} \in W$ um custo igual ao comprimento do segmento $\overline{ij}$, verificamos que o problema de determinar a TCM de um conjunto P de pontos é equivalente ao problema de determinar um conjunto independente maximal de custo mínimo em G_P^{is}. Por outro lado, se atribuirmos a cada vértice $v_{ij} \in W$ o custo $\overline{c}_{ij}$, definido na Proposição 3.4, verificamos que o problema de determinar a TCM de um conjunto P de pontos é equivalente ao problema de determinar um conjunto independente de custo máximo em G_P^{is}. Infelizmente, ambos os problemas sobre grafos são *NP-difíceis* [HP94, GJ79], sendo que o último, o qual é conhecido como



(a) P e $S(P)$.



(b) G_P^{is}.

Figura 3.1: Exemplo de grafo de interseção de segmentos.

problema do conjunto independente de peso máximo, é um problema clássico em teoria dos grafos.

Seja Q_{CI} o politopo definido pela envoltória convexa dos vetores de incidência dos conjuntos independentes do grafo G. Outra implicação da equivalência entre triangulações planares e conjuntos independentes é a relação entre o politopo Q_{CI} e os politopos Q^s_{TP} e $\tilde{Q}^s_{TP}$. Claramente, o politopo dos conjuntos independentes do grafo de interseção é equivalente ao politopo $\tilde{Q}^s_{TP}$. Logo, toda desigualdade definidora de faceta em Q_{CI} também definirá faceta em $\tilde{Q}^s_{TP}$. Além disso, como $Q^s_{TP} \subset \tilde{Q}^s_{TP}$ toda desigualdade válida para o Q_{CI} também é válida para Q^s_{TP}.

O politopo dos conjuntos independentes tem sido objeto de estudo desde a década de 70. Via de regra, trabalhos que abordam o tema visam identificar classes de desigualdades fortes para o politopo em questão, preferivelmente desigualdades que definam facetas [Pad73, Tro75, Wol76, CC97]. Nas duas seções seguintes apresentamos as classes de desigualdades oriundas destes trabalhos que serão utilizadas como corte no algoritmo *branch-and-cut* proposto nesta dissertação: as desigualdades de clique e as desigualdades de ciclo ímpar. Dentre as classes de desigualdades válidas conhecidas para o politopo dos conjuntos independentes, a opção pela utilização computacional destas duas classes se deve a considerações práticas. A principal consideração se refere à comprovada eficiência computacional apresentada por estas desigualdades quando empregadas na resolução do problema do conjunto independente de custo máximo [NS92].

Antes, porém, de discutirmos as desigualdades de clique e de ciclo ímpar, para finalizar esta seção, apresentamos uma interessante propriedade referente às soluções ótimas da relaxação linear de uma formulação de PI válida para o problema do conjunto independente de custo máximo, e assim válida para o PTCM.

Seja $G = (V, E)$ um grafo simples não-orientado. Se associarmos os elementos de um vetor $x \in \mathbb{R}^{|V|}$ aos vértices de V e a cada vértice $v_i \in V$ um custo $c_i \in \mathbb{R}$, é fácil constatar que a seguinte formulação é válida para o problema do conjunto independente de custo máximo em G:

$$
\begin{array}{lll}
\max & \sum_{v_i \in V} c_i x_i & \\
\text{Sujeito a} & x_i + x_j \leq 1, & \forall\, (v_i, v_j) \in E, \\
 & x_i \in \{0, 1\}, & \forall\, v_i \in V.
\end{array} \tag{3.1}
$$

Para o caso onde $c_i > 0$, para todo $v_i \in V$, Nemhauser e Trotter [NT75] mostraram a validade do resultado abaixo para a relaxação linear de (3.1). A relaxação linear deste problema é obtida ao substituirmos as restrições $x_i \in \{0, 1\}$ pelas desigualdades $x_i \geq 0$ e $x_i \leq 1$.

Teorema 3.1 ([NT75]) *Suponha que x^* é uma solução ótima para a relaxação linear do problema (3.1) e $I = \{v_i \in V : x_i^* = 1\}$. Existe pelo menos um conjunto independente ótimo em G que contém I.*

Como o problema (3.1) é de maximização e todos os custos na função objetivo são positivos, se $J = \{v_j \in V : x_j^* = 0\}$, então temos que todo vértice em J é vizinho de pelo menos um vértice em I, e assim o conjunto independente de peso máximo que contém I não contém nenhum dos vértices de J. Uma aplicação prática deste resultado é que podemos utilizá-lo em uma fase de pré-processamento para reduzir o problema original ao problema de determinar o conjunto independente de peso máximo do grafo induzido por $V \setminus (I \cup J)$.

Finalmente, vale ressaltar que o Teorema 3.1 diz respeito somente à formulação (3.1) e a inserção de outras desigualdades válidas para o politopo dos conjuntos independentes nesta formulação pode invalidá-lo.

3.2.2 As Desigualdades de Clique

A desigualdade da formulação (3.1) é costumeiramente denominada de *desigualdade de aresta*. Claramente, a desigualdade (I) referente a dois segmentos que se interceptam $\overline{ij}$, $\overline{k\ell}$ $\in S(P)$ consiste na desigualdade de aresta referente aos vértices $v_{ij}, v_{k\ell} \in W$. Assim, trataremos estas desigualdades de forma indistinta e chamaremos a ambas de desigualdade de aresta.

Nesta seção apresentamos uma classe de desigualdades que generaliza a propriedade expressa pelas desigualdades de aresta. Trata-se das *desigualdades de clique*. Uma desigualdade de aresta diz que, se dois vértices são adjacentes em G, então eles não podem estar simultaneamente em um mesmo conjunto independente de G. Uma desigualdade de clique diz que, dado um conjunto de vértices $V' \subseteq V$, se dois a dois seus vértices são adjacentes, então no máximo um deles poderá estar em um conjunto independente qualquer de G.

Formalmente, dado um grafo $G = (V, E)$ simples não-orientado, uma *clique* em G consiste em um conjunto de vértices $K \subseteq V$, com $|K| \geq 2$, tal que $v_i, v_j \in K$ somente se $(v_i, v_j) \in E$. Abaixo temos uma demonstração formal de que, se $K \subseteq V$ é uma clique maximal, então a desigualdade de clique referente a K define faceta em Q_{CI}.

Teorema 3.2 ([Pad73]) *Uma desigualdade válida do tipo*

$$\sum_{v_j \in K} x_i \leq 1, \tag{VII}$$

onde $K \subseteq V$, define faceta em Q_{CI} se e somente se K é uma clique maximal em G.

Prova: ($\Rightarrow$) Se K não é uma clique existem pelo menos dois vértices $v_i, v_j \in K$ tal que $(v_i, v_j) \notin E$, e assim a desigualdade não é válida pois o vetor de incidência do conjunto $\{v_i, v_j\}$ está em Q_{CI}. Se K é uma clique, porém não maximal, então existe $v_i \in V \backslash K$ tal que $K \cup \{v_i\}$ é uma clique de cardinalidade maior que K, e assim a face definida pela desigualdade (VII) referente a K está contida na face definida pela desigualdade $x_i \geq 0$.

($\Leftarrow$) Desde que K é uma clique, existe uma aresta $(v_i, v_j) \in E$ para cada par $v_i, v_j \in K$. Conseqüentemente, a desigualdade (VII) é válida. Para completarmos a prova mostramos que existem $|V|$ vetores linearmente independentes que satisfazem (VII) na igualdade. $|K|$ soluções podem ser construídas fazendo-se $x_i = 1$ para exatamente um $v_i \in K$, e $x_j = 0$ caso contrário. Os vetores restantes podem ser obtidos da seguinte forma: para cada $v_i \in V \backslash K$ existe no mínimo um $v_k \in K$ tal que $(v_i, v_k) \notin E$, pois K é uma clique maximal. Conseqüentemente, $x_i = 1, x_k = 1, x_j = 0$, para todo $v_j \in V \backslash \{v_i, v_k\}$, também satisfaz (VII) na igualdade. Como a matriz composta pelos $|V|$ vetores acima é triangular (com uma devida permutação de linhas e colunas) estes vetores são linearmente independentes. □

Note que podemos reescrever a formulação apresentada na Seção 3.1 substituindo as desigualdades de aresta por um conjunto de desigualdades de clique que recubram todas as arestas do grafo de interseção. Esta nova formulação, em princípio, apresenta pelo menos duas vantagens: o número de cliques necessárias para recobrir um grafo é, obviamente, menor ou igual ao número de arestas deste grafo, e além disso, a formulação com as desigualdades de clique é mais forte que a formulação original. No entanto, como veremos no Capítulo 5, pelo menos para os caso onde o grafo em questão é um de grafo de interseção de segmentos, o ganho apresentado pela utilização computacional desta formulação não justificou o esforço extra despendido para gerá-la.

Na Figura 3.2 apresentamos um conjunto de pontos para o qual o grafo de interseção de segmentos apresenta uma clique de cardinalidade maior que dois (uma clique de cardinalidade dois equivale a uma aresta). Representando por W o conjunto de vértices deste grafo, é fácil perceber que o conjunto $\{v_{14}, v_{25}, v_{36}\} \subset W$ é uma clique maximal.

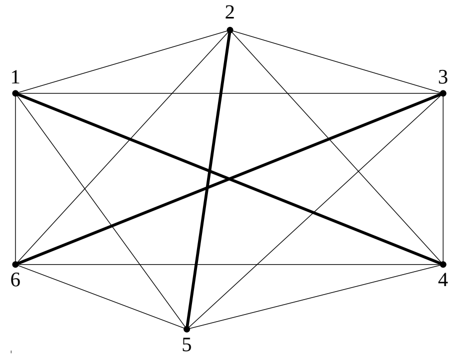


Figura 3.2: Exemplo de um conjunto de pontos cujo grafo de interseção de segmentos possui uma clique de cardinalidade maior que 2.

3.2.3 As Desigualdades de Ciclo Ímpar

Seja $G = (V, E)$ um grafo simples não-orientado. Dizemos que uma seqüência de vértices $V' = \{v_1, v_2, ..., v_k\} \subseteq V$ define um *caminho* se $(v_i, v_{i+1}) \in E$ para todo $i \in \{1, ..., k-1\}$. Se $v_1 = v_k$ dizemos que este caminho forma um *ciclo*. Se não existem arestas entre dois vértices não adjacentes em um ciclo dizemos que este ciclo não possui *cordas*.

O seguinte resultado caracteriza o número máximo de vértices de um ciclo de tamanho ímpar, ou simplesmente *ciclo ímpar*, sem cordas que podem estar simultaneamente em um mesmo conjunto independente de G.

Proposição 3.7 *Seja $C \subseteq V$ um ciclo sem cordas de tamanho $2k+1$, com $k \geq 1$. Então, no máximo k vértices de C podem estar em um mesmo conjunto independente de G.*

Prova: Vamos considerar os conjuntos independentes que contenham um vértice arbitrário $u_i \in C$. Sejam u_j e u_k os vértices adjacentes a u_i em C e seja $C' = C \setminus \{u_i, u_j, u_k\}$. Representado por $\alpha(V')$ o número máximo de vértices de $V' \subseteq V$ que podem estar em um mesmo conjunto independente, temos que $\alpha(C) = \alpha(C') + 1$.

Seja $\{v_1, v_2, ..., v_{2(k-1)}\}$ a seqüência de vértices que C' representa. Considere os conjuntos $\{v_i, v_{i+1}\} \subseteq C'$, onde i é ímpar. Claramente, no máximo um dos vértices de cada um destes conjuntos pode estar em um conjunto independente qualquer de G. Como são $k-1$ conjuntos segue que $\alpha(C') = k-1$ e, conseqüentemente, $\alpha(C) = k$. □

Obviamente, esta proposição garante a validade da seguinte desigualdade para o politopo dos conjuntos independentes:

$$\sum_{v_i \in C} x_i \leq (|C| - 1)/2, \qquad \text{(VIII)}$$

onde C é um ciclo ímpar sem cordas. Costumeiramente, esta desigualdade é denominada de *desigualdade de ciclo ímpar*.

Infelizmente, a desigualdade de ciclo ímpar não define faceta no caso geral. Observe a Figura 3.3, onde os vértices representados por círculos preenchidos formam um ciclo ímpar sem cordas. Note que, a face definida pela desigualdade $x_1 + x_2 + x_3 + x_4 + x_5 \leq 2$ está contida na face definida pela desigualdade $x_6 \geq 0$.

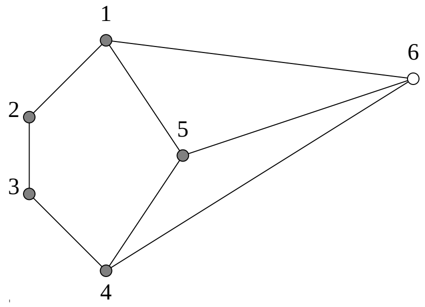


Figura 3.3: Exemplo de ciclo ímpar sem cordas cuja desigualdade correspondente não define faceta em Q_{CI}.

Entretanto, se consideramos, o caso onde o próprio conjunto V define um ciclo ímpar sem cordas, o teorema abaixo assegura que a desigualdade (VIII) define faceta em Q_{CI}.

Teorema 3.3 *Dado um grafo $G = (V, E)$, se V é um ciclo ímpar sem cordas, então a desigualdade (VIII) define faceta em Q_{CI}.*

Prova: Seja F a face definida pela desigualdade (VIII) e seja $\lambda x \leq \lambda_0$ uma desigualdade válida para Q_{CI}. Suponha que

$$F \subseteq F_\lambda = \{x \in Q_{CI} : \lambda x = \lambda_0\}.$$

Como Q_{CI} possui dimensão plena, vamos mostrar que a desigualdade $\lambda x \leq \lambda_0$ é um múltiplo escalar da desigualdade do ciclo ímpar.

Observe que, dado um conjunto independente $I \in V$, se $x^I \in F$, então existem $v_i, v_j, v_k \in V$ tal que: $v_i, v_j \notin I$, $(v_i, v_j) \in E$, $v_k \in I$ e $(v_j, v_k) \in E$. Assim, dado $I' = (I \setminus \{v_k\}) \cup \{v_j\}$ temos que $x^{I'} \in F$.

Por simetria, podemos generalizar a observação acima da seguinte forma. Para quaisquer dois vértices $v_i, v_j \in V$ tal que $(v_i, v_j) \in E$, existem dois conjuntos independentes I e I' tal que $I' = (I \setminus \{v_i\}) \cup \{v_j\}$ e $|I| = |I'| = (|V| - 1)/2$. Logo, como $x^I, x^{I'} \in F \subseteq F_\lambda$, temos que $\lambda_i = \lambda_j := \gamma_{ij}$. Visto que G é conexo, por transitividade, seque que:

$$\lambda_i := \gamma, \quad \forall\, v_i \in V.$$

Por conseguinte,

$$\lambda_0 = \gamma\,(|V| - 1)/2,$$

e assim concluímos a prova. $\square$

Utilizando este resultado, Padberg [Pad73] apresentou um procedimento de *lifting* através do qual é possível identificar, a partir de uma desigualdade de ciclo ímpar, desigualdades que definem faceta em Q_{CI}. Posteriormente, Nemhauser e Trotter [NT74] generalizaram este resultado através do seguinte teorema.

Teorema 3.4 ([NT74]). *Suponha que a desigualdade*

$$\sum_{v_i \in V'} \pi_i x_i \leq \pi_0 \tag{3.2}$$

define faceta em $Q'_{CI} \subseteq Q_{CI}$, onde Q'_{CI} é o politopo dos conjuntos independentes do subgrafo de G induzido por $V' \subset V$. Então existe $\pi_i \in I\!R$ para todo $v_i \in V \backslash V'$ tal que a desigualdade

$$\sum_{v_i \in V} \pi_i x_i \leq \pi_0 \tag{3.3}$$

define faceta em Q_{CI}.

Prova: Devido a recursividade da afirmação, é suficiente provar o teorema para $V' = \{v_1, \ldots, v_{n-1}\}$. Definimos, $\pi_n = \max\{0, \pi_0 - z^*\}$, onde

$$\begin{array}{lll} z^* = & \max \sum_{v_i \in V'} \pi_i x_i & \\ \text{Sujeito a} & \sum_{v_i \in V'} a^i x_i \leq \mathbb{1} - a^n, & \forall\, i \text{ tal que } v_i \in V', \\ & x_i \in \{0,1\}, & \forall\, v_i \in V', \end{array} \tag{3.4}$$

a^i denota a coluna da matriz de incidência do grafo G correspondente ao vértice v_i e $\mathbb{1}$ é um vetor $(1, ..., 1)$ com $|V'|$ elementos. Necessitamos, agora, provar que (3.3) é válida para Q_{CI}, e que é satisfeita na igualdade por n soluções viáveis afim independentes.

Dado um conjunto independente I de G, seja $J = I \cap V'$ e sejam x^I e x^J os vetores de incidência de I e J, respectivamente. Desde que $J \subseteq V'$, é claro que x^J satisfaz (3.2). Se $I = J$, então x^I também satisfaz (3.2). Se $\pi_n > 0$, então $z^* < \pi_0$ e desde que x^J é a solução viável para (3.3) temos que

$$\sum_{v_i \in V'} \pi_i x_i^J \leq z^*.$$

Adicionando-se $\pi_n = \pi_0 - z^*$ a esta inequação vemos que x^I satisfaz (3.3). Logo, (3.3) é uma inequação válida para Q_{CI}.

Desde que (3.2) define uma faceta em Q'_{CI}, existem $n-1$ conjuntos independentes $I_1, \ldots, I_{n-1}$ em V' cujos vetores de incidência $x^1, \ldots, x^{n-1}$ são afim independentes e satisfazem (3.3) na igualdade. Seja $\widehat{x}^n$ a solução ótima de (3.4), a qual corresponde a um conjunto independente $\widehat{I}^n$. Como o lado direito de (3.4) é $\mathbb{1} - a^n$, não existem arestas ligando v_n a vértices em $\widehat{I}_n$, e assim o conjunto $I_n := \widehat{I}_n \cup \{v_n\}$ também é um conjunto independente de G. Desta forma, a definição de π_n força o vetor x^n correspondente a satisfazer (3.3) na igualdade. Ademais, visto que $v_n \in I_n \backslash I_i$ para $1 \leq j \leq n-1$, temos que os vetores $x^1, \ldots, x^n$ são afim independentes, e assim completamos a prova. □

Infelizmente, o procedimento exibido na prova do Teorema 3.4 parece de pouca utilidade prática, visto que o próprio problema (3.4) consiste no problema do conjunto independente de peso máximo e necessita ser resolvido para cada $v_i \in N(V')$, onde $N(V')$ representa o conjunto vértices vizinhos a V'. Além disso, para um dado V' diferentes ordenações dos vértices de $N(V')$ podem produzir diferentes facetas de Q_{CI}.

Ainda assim, o Teorema 3.4 juntamente com o Teorema 3.3 nos garante o seguinte resultado.

Corolário 3.1 ([Pad73]) *Seja $C \subseteq V$ um ciclo ímpar. Então existe pelo menos uma desigualdade*

$$\sum_{v_j \in C} x_j + \sum_{v_j \in V \setminus C} \pi_j x_j \leq (|C| - 1)/2$$

que define faceta em Q_{CI}.

Prova: Segue diretamente dos Teoremas 3.3 e 3.4. □

3.3 As Desigualdades do Conjunto Dominante

Seja $G = (V, E)$ um grafo simples não-orientado. Um *conjunto dominante* de G consiste em um conjunto de vértices $V' \subseteq V$ para o qual, se $v_i \in V \setminus V'$, então existe $v_j \in V'$ tal que $(v_i, v_j) \in E$.

Não é difícil perceber que, assim como o conceito de independência entre segmentos estabelece uma relação entre as triangulações planares de P e os conjuntos independentes de G_P^{is}, a condição de maximalidade do número de segmentos estabelece uma relação entre estas mesmas triangulações e os conjuntos dominantes de G_P^{is}.

Infelizmente, diferentemente do caso dos conjuntos independentes, não encontramos na literatura trabalhos associados ao estudo específico do politopo dos conjuntos dominantes. Contudo, não é difícil constatar que a desigualdade abaixo não só é válida para este politopo, como é suficiente para caracterizar os vetores de incidência dos conjuntos dominantes do grafo G:

$$x_i + \sum_{v_j \in N(v_i)} x_j \geq 1, \quad \forall\, v_i \in V, \tag{IX}$$

onde $N(v_i)$ representa o conjunto de vértices adjacentes ao vértice v_i em G. Esta desigualdade diz que se um vértice não pertence a um certo conjunto dominante de G, então pelo menos um de seus vizinhos deve pertencer. Se o grafo em questão é um grafo de interseção de segmentos, então a desigualdade diz que se um segmento não está em uma triangulação planar, então pelo menos um dos segmentos que o intercepta deve estar.

Claramente, a desigualdade (IX), a qual denominamos *desigualdade do conjunto dominante*, é válida para Q^s_{TP}, mas não é para $\tilde{Q}^s_{TP}$. Porém, dado um grafo de interseção de segmentos $G^{is}_P = (W, H)$, visto que todo $x \in Q^s_{TP}$ satisfaz a equação (II), podemos reescrevê-la como

$$\sum_{v_{k\ell} \notin N(v_{ij}) \cup \{v_{ij}\}} x_{k\ell} \leq \phi_s(P) - 1, \quad \forall\, v_{ij} \in W. \tag{X}$$

Não é difícil perceber que a desigualdade (X) também é válida para $\tilde{Q}^s_{TP}$.

Dado um segmento $\overline{ij} \in S(P)$, seja $I(\overline{ij})$ o conjunto de todos os segmentos em $S(P) \setminus \{\overline{ij}\}$ que o interceptam. Observe que, $\phi_s(P) - 1$ representa o número máximo de segmentos em $S(P) \setminus (I(\overline{ij}) \cup \{\overline{ij}\})$ que dois a dois não se interceptam. Se representarmos por F a face definida pela desigualdade (X) referente a $\overline{ij}$ em $\tilde{Q}^s_{TP}$, vemos que, se uma equação válida para Q^s_{TP} é decorrente da propriedade de maximalidade e seus coeficientes diferentes de zero estão associados somente aos segmentos em $S(P) \setminus (I(\overline{ij}) \cup \{\overline{ij}\})$, então esta equação também é válida para F.

Desta forma, se tais equações existem, temos que a desigualdade obtida de (X) pela subtração destas equações também será válida para $\tilde{Q}^s_{TP}$ e Q^s_{TP}. Claramente, a desigualdade resultante deste processo é mais forte que a desigualdade (X), com respeito a $\tilde{Q}^s_{TP}$. Contudo, não é difícil perceber que ela define em Q^s_{TP} a mesma face que a desigualdade (X), e conseqüentemente que a desigualdade (IX). Assim, podemos utilizar esta nova desigualdade para avaliar a força da desigualdade (IX).

Abaixo apresentamos duas classes de equações que satisfazem os requisitos mencionados acima e as utilizamos para fortalecer a desigualdade (X) com respeito a $\tilde{Q}^s_{TP}$.

Para todo subconjunto $P' \subseteq P$, seja $S_F(P')$ o conjunto de todos os segmentos em $S(P')$ que não interceptam nenhum outro segmento deste mesmo conjunto. Pela condição de maximalidade, se T é uma triangulação planar de P, então $S_F(P) \subseteq T$. Em outras palavras para qualquer $x \in Q^s_{TP}$, se $\overline{k\ell} \in S_F(P)$, então $x_{k\ell} = 1$. Assim, dado $\overline{ij} \in S(P)$, com $I(\overline{ij}) = \emptyset$, temos que a desigualdade (X) referente a $\overline{ij}$ consiste na combinação linear da desigualdade abaixo com as equações $x_{ij} = 1$, para todo $\overline{ij} \in S_F(P)$:

$$\sum_{v_{k\ell} \notin N(v_{ij}) \cup V_F \cup \{v_{ij}\}} x_{k\ell} \leq \phi_s(P) - |S_F(P)| - 1, \quad \forall\, v_{ij} \in W, \tag{XI}$$

onde $V_F = \{v_{ij} \in W : \overline{ij} \in S_F(P)\}$.

Suponha a existência de um vértice $v_{ij} \in W$ tal que o conjunto $N(v_{ij}) \cup \{v_{ij}\}$ defina uma clique. Pela desigualdade de clique, no máximo um destes vértices pode estar em um dado conjunto independente de G_P^{is}. Contudo, se nenhum dos vértices em $N(v_{ij})$ estiverem neste conjunto independente, então para que ele seja maximal v_{ij} deve estar. Logo, qualquer que seja o conjunto independente maximal de G_P^{is}, exatamente um dos vértices da referida clique deve estar neste conjunto, i.e, $x_{ij} + \sum_{v_{k\ell} \in N(v_{ij})} x_{k\ell} = 1$. Subtraindo estas equações da desigualdade (XI) obtemos a desigualdade

$$\sum_{v_{k\ell} \notin N(v_{ij}) \cup V_F \cup V_C \cup \{v_{ij}\}} x_{k\ell} \leq \phi_s(P) - |S_F(P)| - k - 1, \quad \forall\, v_{ij} \in W, \tag{XII}$$

onde V_C representa o conjunto de vértices que pertencem a cliques tal com a descrita acima e k é o número destas cliques. A título de exemplo, considere a Figura 3.4, onde $S_F(P) = \{\ \overline{12}\ ,\ \overline{16}\ ,\ \overline{23}\ ,\ \overline{34}\ ,\ \overline{45}\ ,\ \overline{56}\ \}$ e as cliques que obedecem a propriedade descrita acima são $\{v_{17}, v_{26}\}$ e $\{v_{15}, v_{67}\}$. Note que mesmo a desigualdade (XII) referente ao segmento $\overline{24}$ $(x_{14} + x_{25} + x_{27} + x_{46} + x_{47} + x_{57} \leq 3)$, não define faceta em $\tilde{Q}_{TP}^s$ visto que é uma combinação linear das desigualdades de aresta $x_{25} + x_{47} \leq 1$, $x_{14} + x_{27} \leq 1$ e $x_{46} + x_{57} \leq 1$.

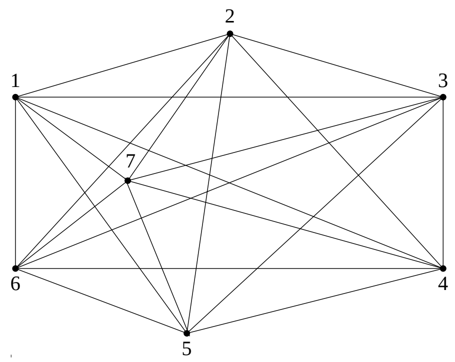
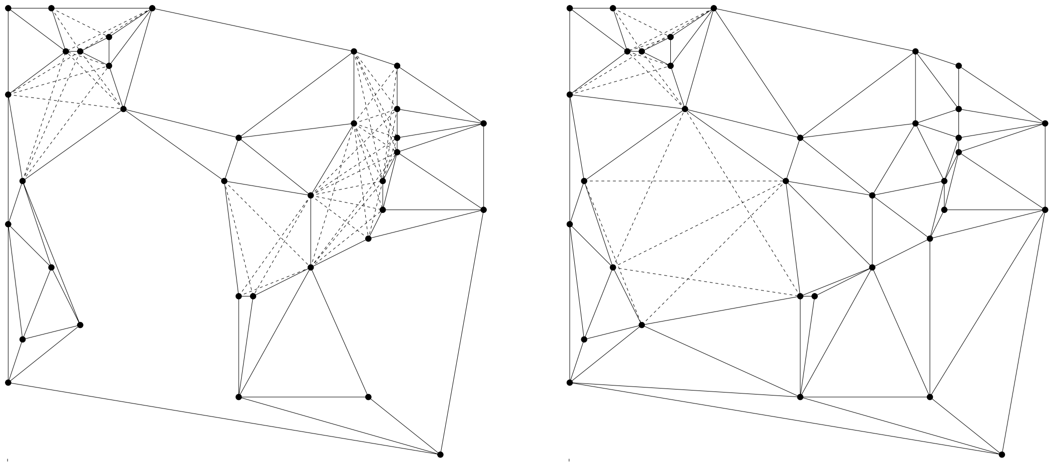


Figura 3.4: Exemplo no qual a desigualdade (XI) não define faceta em $\tilde{Q}_{TP}^s$.

Apesar de não definir faceta no caso geral, pudemos constatar a eficiência computacional da desigualdade do conjunto dominante. Considere a Figura 3.5, onde temos duas soluções fracionárias ótimas para a mesma instância. A solução (a) foi obtida utilizando as desigualdades (I)–(IV) e na obtenção da solução (b) acrescentamos, às desigualdades (I)–(IV), as desigualdades (IX). Nesta figura os segmentos cheios representam as variáveis que assumiram o valor 1 na relaxação, enquanto os segmentos tracejados representam variáveis que assumiram valores fracionários (entre 0 e 1). Observe que a solução (a) possui áreas com grande concentração de segmentos cujas variáveis correspondentes estão no suporte da solução ($x_{ij} > 0$), e outras onde esta concentração é muito baixa. A solução (b) mostra que a inserção das desigualdades do conjunto dominante diminui esta diferença, impondo um número mínimo deste tipo de segmentos nas áreas onde antes havia



(a) Relaxação linear da formulação com as restrições (I)–(IV). Custo: 580,59, *gap:* $4,31\%$.

(b) Relaxação linear da formulação com as restrições (I)–(IV) e (IX). Custo: 604,39, *gap:* $0,39\%$.

Figura 3.5: Duas soluções fracionárias ótimas para o PTCM (custo da solução inteira ótima: 606,78).

uma baixa concentração de segmentos. Isto, como podemos ver, reduz significativamente o *gap de integralidade*, i.e., a diferença percentual entre o custo da solução inteira ótima e o custo da solução da relaxação linear.

3.4 As Desigualdades de Planaridade

As desigualdades apresentadas nas últimas duas seções são baseadas em propriedades válidas para conjuntos independentes maximais em um grafo qualquer. Nesta seção tratamos de uma classe de desigualdades que expressam uma propriedade inerentemente geométrica das triangulações planares. Trata-se de uma generalização da desigualdade (VI), vista anteriormente.

Como mencionado, a cardinalidade de qualquer triangulação planar $T \subseteq S(P)$ é $\phi_s(P)$. Isto significa que qualquer conjunto $S' \subseteq S(P)$, com $|S'| > \phi_s(P)$, não pode representar uma figura planar. Como isto é válido para qualquer conjunto finito P' de pontos no plano, com $|P'| \geq 3$, esta propriedade nos leva a uma nova desigualdade válida para $\tilde{Q}^s_{TP}$. Seja $P' \subseteq P$ um conjunto de pontos no plano tal que $|P'| \geq 3$. Toda triangulação planar de P satisfaz à desigualdade

$$\sum_{\overline{ij} \in S(P')} x_{ij} \leq \phi_s(P'). \tag{XIII}$$

Dado $S' \subseteq S(P')$, temos que $x^{S'}$ pertence à face definida pela desigualdade (XIII) se e somente se S' é uma triangulação planar de P'. Considere o conjunto de segmentos $S_F(P')$, definido na seção anterior (desigualdades do conjunto dominante). Como $S_F(P') \subseteq S(P)$ está contido em toda triangulação planar de P', segue que o número máximo de segmentos em $S(P') \setminus S_F(P')$, que dois a dois não se interceptam, é $\phi_s(P') - |S_F(P')|$. Logo, a desigualdade (XIII) é uma combinação linear das desigualdades $x_{ij} \leq 1$, para todo $\overline{ij} \in S_F(P')$, com a desigualdade

$$\sum_{\overline{ij} \in S(P') \setminus S_F(P')} x_{ij} \leq \phi_s(P') - |S_F(P')| \tag{XIV}$$

Em outras palavras, a desigualdade (XIII) é dominada, ou é mais fraca que, a desigualdade (XIV).

Infelizmente, no caso geral, a desigualdade (XIV) não define faceta em $\tilde{Q}^s_{TP}$, como podemos verificar no exemplo da Figura 3.6. A desigualdade (XIV) referente ao conjunto de pontos $P' := \{1, 2, 3, 4, 5, 6\}$ é dada por $x_{13}+x_{14}+x_{15}+x_{24}+x_{25}+x_{26}+x_{35}+x_{36}+x_{46} \leq 6$. No entanto, é fácil constatar que a face definida por esta desigualdade está contida na face definida pela desigualdade $x_{78} \geq 0$.

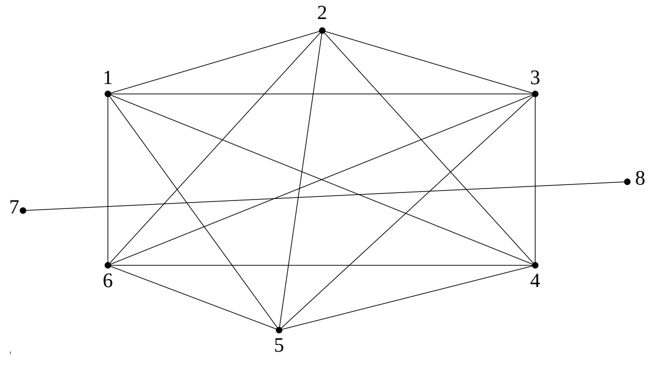


Figura 3.6: Exemplo no qual a desigualdade (XIV) não define faceta em $\tilde{Q}^s_{TP}$.

Note que no exemplo do parágrafo anterior, se a desigualdade em questão definir faceta no politopo monótono das triangulações planares de P', então, utilizando o procedimento exibido na prova do Teorema 3.4, podemos realizar um *lifting* dela para uma desigualdade que defina faceta em $\tilde{Q}^s_{TP}$. Assim, consideremos apenas o caso onde $P' = P$.

Quando $P' = P$, de forma análoga ao que acontece com a desigualdade (X), temos que toda equação válida para Q^s_{TP}, cujos coeficientes diferentes de zero estão associados somente aos segmentos em $S(P) \backslash S_F(P)$, também é válida para a face definida pela desigualdade (XIV). Por exemplo, a face definida pela desigualdade (XIV) referente ao conjunto de pontos da Figura 3.4 também está contida na face definida pela desigualdade $x_{17} + x_{26} \leq 1$. Porém, se além de $P' = P$, exigirmos que todo ponto em P esteja na fronteira de sua envoltória convexa, o teorema abaixo garante que a desigualdade (XIV) define faceta em $\tilde{Q}^s_{TP}$.

Teorema 3.5 *Seja P um conjunto finito de pontos no plano. Se todos os pontos de P são vértices de sua envoltória convexa, então a desigualdade (XIV) referente a P define faceta em $\tilde{Q}^s_{TP}$.*

Prova: Seja F a face definida pela desigualdade (XIV) e seja $\lambda x \leq \lambda_0$ uma desigualdade válida para Q_{CI}. Suponha que

$$F \subseteq F_\lambda = \{x \in Q_{CI} : \lambda x = \lambda_0\}.$$

Vamos mostrar que a desigualdade $\lambda x \leq \lambda_0$ é um múltiplo escalar da desigualdade (XIV).

Seja T uma triangulação planar de P tal que $x^T \in F$. Obviamente, $S_F(P) \subseteq T$. Assim, para cada segmento $\overline{ij} \in S_F(P)$, defina $T_{ij} := T \backslash \{ \overline{ij} \}$. Como $x^T, x^{T_{ij}} \in F \subseteq F_\lambda$, temos que

$$\lambda_{ij} = 0, \quad \forall \ \overline{ij} \in S_F(P). \tag{3.5}$$

Dados dois segmentos quaisquer $\overline{ij}$, $\overline{k\ell} \in S(P) \backslash S_F(P)$, considere agora os seguintes casos:

Caso 1. $\overline{ij}$ e $\overline{k\ell}$ se interceptam.

Seja $T \ni \{ \overline{ij}, \overline{ik}, \overline{jk}, \overline{j\ell}, \overline{i\ell} \}$ uma triangulação planar de P. Note que, devido a disposição dos pontos, o quadrilátero $ikj\ell$ não contém outros pontos de P em seu interior, e assim o conjunto $T' := (T \backslash \{ \overline{ij} \}) \cup \{ \overline{k\ell} \}$ também é uma triangulação planar de P. Logo, como $x^T, x^{T'} \in F \subseteq F_\lambda$, temos que $\lambda_{ij} = \lambda_{k\ell} := \gamma_{ijk\ell}$.

Caso 2. $\overline{ij}$ e $\overline{k\ell}$ não se interceptam.

Dado um segmento de reta $s \in I\!R^2$, sejam L^+_s e L^-_s os dois semi-planos definidos pela reta que contém s. Em função da disposição dos pontos, podemos supor, sem perda de generalidade, que $\overline{ij} \subset L^-_{k\ell}$ e $\overline{k\ell} \subset L^+_{ij}$. Como tanto $\overline{ij}$ quanto $\overline{k\ell}$ não limitam a envoltória convexa de P então devem existir $p, q \in P$ tal que $p \in L^-_{ij}$ e $q \in L^+_{k\ell}$. Novamente em decorrência da disposição dos pontos, temos que o polígono $ipjkq\ell$ é convexo (Figura 3.7). Assim, não é difícil perceber que o segmento $\overline{pq}$ intercepta tanto $\overline{ij}$ quanto $\overline{k\ell}$, e do caso 1 segue que $\lambda_{ij} = \lambda_{pq} = \lambda_{k\ell} := \gamma_{ijk\ell}$.

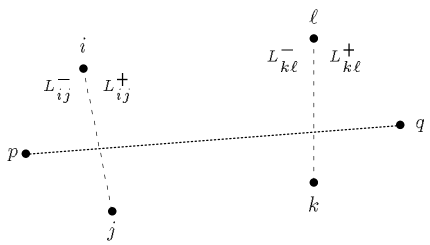


Figura 3.7: Ilustração do caso 2 da prova do Teorema 3.5.

Dos dois casos acima é imediato que

$$\lambda_{ij} := \gamma, \quad \forall \ \overline{ij} \in S(P) \setminus S_F(P). \tag{3.6}$$

Como $F \subseteq F_\lambda$, de (3.5) e (3.6) segue o que queríamos provar. □

Uma implicação deste teorema é que quando o conjunto de pontos P está disposto tal descrito acima temos que $dim(Q^s_{TP}) = |S(P) \setminus S_F(P)|$. Outra conseqüência imediata é o seguinte resultado.

Corolário 3.2 *Seja $P' \subseteq P$ um conjunto finito de pontos no plano. Se todos os pontos de P' são vértices de sua envoltória convexa, então existe pelo menos uma desigualdade*

$$\sum_{\overline{ij} \in S(P')} x_{ij} + \sum_{\overline{ij} \in S(P) \setminus S(P')} \pi_{ij} x_{ij} \leq \phi_s(P') - |S_F(P')|$$

que define faceta em Q_{CI}.

Prova: Segue diretamente dos Teoremas 3.5 e 3.4. □

Apesar do fato da desigualdade (XIV) não definir faceta no caso geral, assim como acontece com as desigualdade de conjunto dominante, ela se mostra muito eficiente na prática. Para exemplificar a atuação desta classe de desigualdades, apresentamos na Figura 3.8 outra solução fracionária ótima para a mesma instância considerada na Figura

3.5. Esta solução foi obtida utilizando a mesma formulação que gerou a solução (b) daquela figura acrescida da desigualdade (XIV) para o conjunto P' representado por círculos brancos.

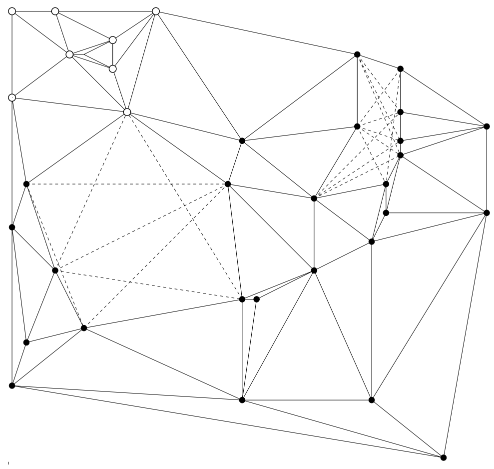


Figura 3.8: Uma solução fracionária ótima com uma desigualdade de planaridade (indicada pelos círculos brancos). Custo: 605,04, *gap:* $0,28\%$.

3.5 Outras Desigualdades Válidas

Para encerrarmos este capítulo, vale ressaltar que outras classes de desigualdades válidas para o politopo das triangulações planares foram identificadas no transcorrer deste trabalho (c.f. [NdS96]). Tais desigualdades não serão descritas aqui por serem consideradas de pouca relevância. Por um lado, do ponto de vista teórico, como elas não definem facetas, não são significativas para a descrição de $\tilde{Q}^s_{TP}$. Por outro lado, do ponto de vista prático, elas não se mostraram computacionalmente eficientes.

Capítulo 4

O Modelo Baseado em Triângulos

Neste capítulo realizamos um estudo complementar ao realizado no Capítulo 3. Ou seja, estudamos um modelo de PI para o PTCM no qual, através de uma associação entre suas variáveis e os triângulos em $\triangle(P)$, visa-se determinar o conjunto de triângulos que compõem uma TCM. De forma análoga ao que foi feito naquele capítulo, chamamos o modelo aqui apresentado de *modelo baseado em triângulos*.

O capítulo encontra-se dividido em 3 partes distintas.

Na Seção 4.1 mostramos que toda desigualdade válida para o modelo baseado em segmentos apresentada no capítulo anterior pode ser reescrita em termos de triângulos. Além disso, mostramos que, assim como acontece naquele capítulo um subconjunto destas desigualdades é suficiente para garantir uma formulação válida para o PTCM.

Contra a utilização computacional da formulação apresentada na Seção 4.1 pesa o fato desta ser estruturalmente idêntica à formulação apresentada na 3.1 e possuir um número significativamente maior de variáveis e restrições. Assim, nas Seções 4.2 e 4.3, baseados no trabalho de Loera et. al [dLHSS96], discutimos duas outras formulações válidas para o PTCM cujas variáveis também estão associadas aos triângulos em $\triangle(P)$. Dentre os resultados apresentados por estas formulações destaca-se aquele que garante que se todos os pontos de P estão sobre a fronteira de sua envoltória convexa, então o politopo definido pelas relaxações lineares destas formulações é equivalente à envoltória convexa das soluções inteiras. Além disso, como veremos no próximo capítulo, foi utilizando uma destas formulações que obtivemos nosso melhor resultado computacional.

Finalmente, na Seção 4.4 apresentamos uma nova classe de desigualdades que expressam uma propriedade que não é satisfeita por todas as triangulações planares, mas somente pelas triangulações minimamente locais, uma classe especial de triangulações planares da qual, como vimos no Capítulo 2, a TCM faz parte.

4.1 Semelhanças entre os Modelos

Como descrito no Capítulo 1, as triangulações planares, e por conseguinte a TCM, podem ser definidas da seguinte forma. Dados dois triângulos em $\triangle(P)$ diremos que estes se interceptam caso a região formada pelo conjunto de pontos na interseção de seus interiores tenha dimensão igual a 2. Uma triangulação planar de P é um conjunto maximal de triângulos vazios com extremos em P que não se interceptam. Uma TCM de P consiste em uma triangulação planar de P cuja soma dos comprimentos dos segmentos de reta definidos pelos lados de seus triângulos vazios é mínimo dentre todas as triangulações planares de P.

Note que, existe uma grande semelhança na forma como o problema foi definido sobre segmentos e sobre triângulos, pois em ambas as abordagens o problema consiste em determinar um conjunto independente maximal de elementos de um outro conjunto. De fato, também a versão sobre triângulos pode ser modelada como uma versão restrita do problema do conjunto independente em um grafo.

Dizemos que $G_P^{it}(W, H)$ é o *grafo das interseções de triângulos* de P se: para cada triângulo $\triangle_{k\ell m} \in \triangle(P)$ existe um vértice $v_{k\ell m} \in W$ que o representa, e para cada dois triângulos $\triangle_{k\ell m}$ e $\triangle_{nop}$ que se interceptam existe uma aresta $(v_{k\ell m}, v_{nop}) \in H$. Assim, determinar um conjunto de triângulos que definam uma triangulação planar de um conjunto P de pontos também é equivalente a determinar um conjunto independente maximal no grafo das interseções de triângulos de P.

Resta, para que possamos representar o PTCM através de PI, definir como atribuir custo aos triângulos vazios de $\triangle(P)$ em termos de seus segmentos (lados). A cada triângulo $\triangle_{k\ell m}$, cujos lados são os segmentos $\overline{k\ell}$, $\overline{km}$ e $\overline{\ell m}$, atribuímos um custo $c_{k\ell m} = \alpha_{k\ell} c_{k\ell} + \alpha_{km} c_{km} + \alpha_{\ell m} c_{\ell m}$, onde c_{ij} é o comprimento do segmento $\overline{ij}$ e $\alpha_{ij} = 1$ se o segmento $\overline{ij}$ limita a fronteira da envoltória convexa de P, e $1/2$ caso contrário. Note que, dada uma triangulação planar, os segmentos desta triangulação que limitam a envoltória convexa de P aparecem como lado de um único triângulo vazio, enquanto o restante dos segmentos aparecem como lado de dois triângulos. Assim, o custo de uma triangulação medido através da soma dos custos de seus triângulos é igual a soma dos comprimentos de segmentos induzidos por estes triângulos.

Podemos então formular o PTCM através de PI da seguinte forma: a cada triângulo $\triangle_{k\ell m} \in \triangle(P)$, cujo custo é representado por $c_{k\ell m}$, associa-se uma variável $x_{k\ell m} \in \{0, 1\}$. Seja x o vetor das variáveis $x_{k\ell m}$. É possível garantir que o valor de $x_{k\ell m}$ será 1 se e somente se $\triangle_{k\ell m}$ pertencer à triangulação planar representada por x através da seguinte formulação:

$$
\begin{array}{lll}
\min & \displaystyle\sum_{\triangle k\ell m \in \triangle(P)} c_{k\ell m} x_{k\ell m} & \\
\text{Sujeito a} & x_{k\ell m} + x_{nop} \leq 1, & \forall \triangle_{k\ell m}, \triangle_{nop} \in \triangle(P) \\
 & & \text{tal que } \triangle_{k\ell m} \cap \triangle_{nop} \neq \emptyset, \qquad \text{(I)} \\
 & \displaystyle\sum_{\triangle_{k\ell m} \in \triangle(P)} x_{k\ell m} = 2(n-1) - h, & \qquad\qquad\qquad \text{(II)} \\
 & x_{k\ell m} \in \{0, 1\}, & \forall \triangle_{k\ell m} \in \triangle(P).
\end{array}
$$

onde h representa o número de pontos de P sobre a fronteira de sua envoltória convexa. A desigualdade (I) diz que dois triângulos de uma mesma triangulação planar não devem se interceptar. A restrição (II), a qual é obtida da Proposição 3.1 e da fórmula de Euler, determina o número exato de triângulos em qualquer triangulação planar de um dado conjunto de pontos. Esta última restrição é utilizada para garantir a condição de maximalidade do número de triângulos.

Note que a formulação acima é estruturalmente idêntica à formulação da Seção 3.1, i.e., em ambas as formulações o vetor solução é o vetor de incidência de um conjunto independente maximal em um certo grafo. Assim, todos os resultados existentes para o problema do conjunto independente em um grafo utilizados na abordagem baseada em segmentos, a exemplo das desigualdades de aresta, clique e ciclo ímpar, e mesmo a desigualdade do conjunto dominante, podem ser utilizados nesta abordagem. Além disso, note que, como indica a equação (II) a desigualdade de planaridade, apresentada no Capítulo 3, também pode ser reescrita em termos de triângulos.

Assim, os principais resultados (desigualdades) válidos para a formulação baseada em segmentos também são válidos para a formulação baseada em triângulos. Contudo, enquanto o número de variáveis e restrições utilizadas na formulação da Seção 3.1 são limitados, respectivamente, por $O(n^2)$ e $O(n^4)$, estes mesmos números na formulação acima são limitados por $O(n^3)$ e $O(n^6)$. Ainda assim, o modelo baseado em triângulos nos oferece outras possibilidades na representação do PTCM através de PI. O restante deste capítulo é dedicado a discutir tais possibilidades.

4.2 A Formulação Baseada em Partição de Conjuntos

Uma triangulação planar consiste em uma partição da envoltória convexa de P em regiões limitadas por triângulos vazios com extremos em P. Isto significa que, dado um conjunto de triângulos vazios com extremos em P, estes triângulos compõem uma triangulação planar se e somente se todo ponto em $conv(P)$ está contido no interior de exatamente um destes triângulos. Claramente, esta constatação garante a validade da equação

$$\sum_{\triangle_{ijk} \ni q} x_{ijk} = 1, \quad \forall\, q \in conv(P) \setminus P.$$

Seja $R(P)$ o conjunto das menores regiões limitadas pelos segmentos em $S(P)$ (veja o exemplo da Figura 4.1). Seja $r \in R(P)$ e sejam $p, q \in r$. Observe que a equação acima referente a p é idêntica àquela referente a q. Assim, podemos reescrever o conjunto de equações acima da seguinte forma:

$$\sum_{\triangle_{ijk} \supseteq r} x_{ijk} = 1, \quad \forall\, r \in R(P). \tag{III}$$

Esta classe de equações foi proposta originalmente por Loera et. al [dLHSS96] e não é difícil constatar que ela é suficiente para caracterizar os vetores de incidência das triangulações planares de um dado conjunto de pontos. Desta forma, uma outra formulação de PI possível para o PTCM é:

$$(\text{PC}) \qquad \begin{array}{lll} \min & \displaystyle\sum_{\triangle_{k\ell m} \in \triangle(P)} c_{k\ell m} x_{k\ell m} & \\ \text{Sujeito a} & \displaystyle\sum_{\triangle_{ijk} \supseteq r} x_{ijk} = 1, & \forall\, r \in R(P), \\ & x_{k\ell m} \in \{0, 1\} & \forall\, \triangle_{k\ell m} \in \triangle(P). \end{array}$$

Note que a formulação acima representa o PTCM como um problema de *partição de conjuntos*. Isto representa um fato positivo, visto que formulações com esta estrutura costumam ser "fortes". Por outro lado, gerar todas as equações da classe (III) representa um problema computacional um tanto complexo. Note que, dependendo da configuração dos pontos em questão, determinar computacionalmente as regiões que compõem $R(P)$ pode ser um problema extremamente sujeito a erros de imprecisão numérica.

Devido ao problema de ordem prática mencionado no parágrafo anterior, na próxima seção apresentamos uma outra formulação de PI para o PTCM que não apresenta tais inconvenientes. Mais que isto, mostramos que esta nova formulação é equivalente a (PC) e que sob certa circunstância a relaxação linear desta formulação equivale à envoltória convexa das soluções inteiras do problema. Para que possamos demonstrar este último resultado considere o lema abaixo, onde denominamos por *suporte* de um vetor $x \in$

$I\!R^{|\triangle(P)|}$, e representamos por $sup(x)$, o conjunto de triângulos $\triangle_{k\ell m} \in \triangle(P)$ para os quais $x_{k\ell m} \neq 0$.

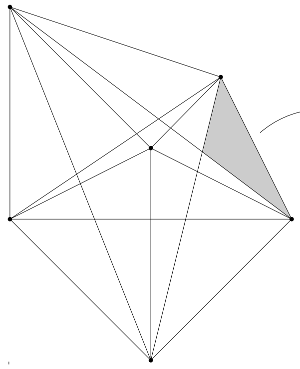


Figura 4.1: Exemplo de regiões limitadas.

Lema 4.1 ([dLHSS96]) *Seja Q o politopo definido pela relaxação linear de (PC). Um vértice x de Q está em coordenadas inteiras se e somente se contém uma triangulação planar em seu suporte.*

Prova: ($\Rightarrow$) Trivial.

($\Leftarrow$) Se $sup(x)$ é exatamente uma triangulação planar de P, então as equações (III) garantem que x é o vetor de incidência desta triangulação. Suponhamos, então, que existe uma triangulação planar $T \subset \triangle(P)$ tal que $sup(x) \supset T$. Neste caso, o vetor $x_\epsilon := \frac{x - \epsilon x^T}{1-\epsilon}$, para algum escalar positivo ϵ suficientemente pequeno, também pertence a Q, o que implica que $x = (1-\epsilon)x_\epsilon + \epsilon x^T$ não é um vértice de Q. $\Box$

4.3 A Formulação Baseada nas Igualdades do Cocircuito

Dada uma triangulação planar de P, todo segmento nesta triangulação que não limita a fronteira da envoltória convexa de P é lado de exatamente dois triângulos. Cada um destes triângulos está localizado em um dos semi-planos definidos pela reta que contém o segmento (conforme pode ser observado na Figura 1.1). Além disso, se um certo segmento não está nesta triangulação, obviamente, nenhum dos triângulos que o possuem como lado deverá ser formado. Ou seja, se $L^+_{k\ell}$ e $L^-_{k\ell}$ são os semi-planos definidos pela reta que contém o segmento $\overline{k\ell}$, o número de triângulos em $L^+_{k\ell}$ que possuem $\overline{k\ell}$ como lado é igual ao número de triângulos em $L^-_{k\ell}$ que também possuem $\overline{k\ell}$ como lado. Esta propriedade das triangulações planares pode ser expressa na forma de uma equação linear da seguinte forma:

$$\sum_{\substack{\triangle_{k\ell m} \in \triangle(P), \\ \triangle_{k\ell m} \subset L^+_{k\ell}}} x_{k\ell m} - \sum_{\substack{\triangle_{k\ell n} \in \triangle(P), \\ \triangle_{k\ell n} \subset L^-_{k\ell}}} x_{k\ell n} = 0, \quad \forall\ \overline{k\ell} \in S(P) \backslash S_H(P), \tag{IV}$$

onde $S_H(P) \subseteq S(P)$ é o conjunto de segmentos que limitam a envoltória convexa de P.

Como um segmento que limita a envoltória convexa de P sempre é lado de algum triângulo em qualquer triangulação planar, temos que a seguinte equação também é válida para o politopo das triangulações planares:

$$\sum_{\substack{\triangle_{k\ell m} \in \triangle(P), \\ \triangle_{k\ell m} \supset \{k,\ell\}}} x_{k\ell m} = 1, \quad \forall\ \overline{k\ell} \in S_H(P). \tag{V}$$

Também estas duas equações foram apresentadas originalmente por Loera et. al [dLHSS96]. Como estas equações também representam propriedades da *matróide orientada* associada ao conjunto de pontos em questão, os autores as denominaram de equação do *cocircuito interno* e do *cocircuito externo*, respectivamente. Podemos utilizar tais equações para formular o PTCM da seguinte forma:

$$(\mathrm{Co}) \quad \begin{array}{lllll} \min & \displaystyle\sum_{\triangle_{k\ell m} \in \triangle(P)} c_{k\ell m} x_{k\ell m} & & & \\ \text{Sujeito a} & \displaystyle\sum_{\substack{\triangle_{k\ell m} \in \triangle(P), \\ \triangle_{k\ell m} \subset L^+_{k\ell}}} x_{k\ell m} - & \displaystyle\sum_{\substack{\triangle_{k\ell n} \in \triangle(P), \\ \triangle_{k\ell n} \subset L^-_{k\ell}}} x_{k\ell n} = 0, & & \forall\ \overline{k\ell} \in S(P) \backslash S_H(P), \\ & \displaystyle\sum_{\substack{\triangle_{k\ell m} \in \triangle(P), \\ \triangle_{k\ell m} \supset \{k,\ell\}}} x_{k\ell m} = 1, & & & \overline{k\ell} \in S_H(P), \\ & x_{k\ell m} \in \{0,1\}, & & & \forall \triangle_{k\ell m} \in \triangle(P). \end{array}$$

O seguinte resultado garante a equivalência entre (PC) e (Co).

Teorema 4.1 ([dLHSS96]) *As formulações (PC) e (Co) são equivalentes, i.e., dado um vetor* $x \in \mathbb{R}^{|\triangle(P)|}$, *x satisfaz as igualdades de (PC) se e somente se satisfaz às igualdades de (Co).*

Prova:[1] ($\Rightarrow$) Note que todo segmento $\overline{ij} \in S_H(P)$ limita uma única região $r \in R(P)$ e que r é coberta apenas por triângulos que possuem $\overline{ij}$ como lado. Logo, a equação (V) referente ao segmento $\overline{ij}$ é idêntica a equação (III) referente à região r. Consideremos agora a equação (IV) referente ao segmento $\overline{ij} \in S(P) \setminus S_H(P)$. Sejam $r_1, r_2 \in R(P)$ duas regiões *vizinhas* limitadas pelo segmento $\overline{ij}$, onde dizemos que duas regiões são vizinhas se o conjunto de pontos definido pela interseção destas regiões possui dimensão igual a 1 (Figura 4.2). Observe que, dado um triângulo $\triangle_{k\ell m} \in \triangle(P)$, se $\triangle_{k\ell m} \supset r_1$ não possui o lado $\overline{ij}$, então $r_2 \subset \triangle_{k\ell m}$. Assim, podemos obter a equação (IV) referente ao segmento $\overline{ij}$ subtraindo a equação (III) referente à região r_1 da equação (III) referente à região r_2.

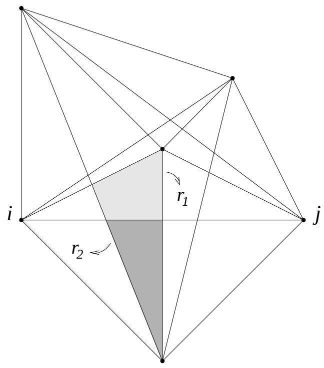


Figura 4.2: Exemplo de regiões vizinhas.

($\Leftarrow$) Primeiro vamos mostrar que os triângulos em $sup(x)$ cobrem toda envoltória convexa de P.

Suponha que x satisfaça as equações de (Co). Obviamente, pela equação (V), pelo menos parte da envoltória convexa de P deve ser coberta pelos triângulos em $sup(x)$. Contudo, por contradição, suponha que existam regiões dentro da envoltória convexa de P que não estão sendo cobertas por estes triângulos. Claramente, estas regiões são separadas das regiões cobertas por polilinhas (abertas ou fechadas). Assim, existe pelo menos um triângulo $\triangle_{k\ell m} \in sup(x)$, tal que pelo menos um de seus lados é um dos segmentos da polilinha que separa a região que ele ajuda a cobrir de uma região não coberta (veja Figura 4.3). Sem perda de generalidade, suponha que $\overline{k\ell}$ seja este segmento e que $\triangle_{k\ell m} \subset L^+_{k\ell}$. Como $\overline{k\ell}$ está na fronteira entre uma região coberta e uma não coberta, não existe nenhum triângulo $\triangle_{k\ell n} \subset L^-_{k\ell}$, tal que $x_{k\ell n} \in sup(x)$, e assim a desigualdade (IV) referente ao segmento $\overline{k\ell}$ não está sendo satisfeita. Entretanto, isto contraria a hipótese de que x' satisfaz todas as equações de (Co).

[1] A prova deste teorema tal como apresentada aqui não é encontrada na literatura. A prova apresentada em [dLHSS96] utiliza o conceito de matróides orientadas.

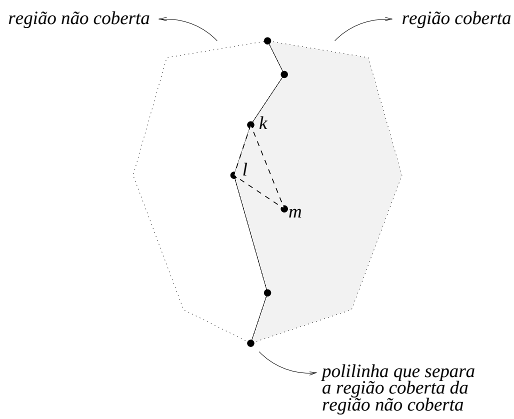


Figura 4.3: Exemplo de um triângulo cujo um dos lados é um dos segmentos de uma polilinha que separa uma região coberta de uma região não coberta.

Seja R_x o conjunto de regiões limitadas formadas pelos segmentos que são lados dos triângulos em $sup(x)$. Sejam $r_1, r_2 \in R_x$ duas regiões vizinhas limitadas pelo segmento $\overline{ij}$. Considere $T_1 := \{\triangle_{k\ell m} \in sup(x) : \triangle_{k\ell m} \supset r_1\}$, $T_2 := \{\triangle_{k\ell m} \in sup(x) : \triangle_{k\ell m} \supset r_2\}$ e $T_\cap := T_1 \cap T_2$. A equação (IV) garante que $\sum_{\triangle_{k\ell m} \in T_1 \setminus T_\cap} x_{k\ell m} = \sum_{\triangle_{k\ell m} \in T_2 \setminus T_\cap} x_{k\ell m}$. Como todo triângulo $\triangle_{k\ell m} \in T_\cap$ contém tanto r_1 quanto r_2 segue que $\sum_{\triangle_{k\ell m} \in T_1} x_{k\ell m} = \sum_{\triangle_{k\ell m} \in T_2} x_{k\ell m}$. Pela equação (V), existe uma região $r \in R(P)$ tal que $\sum_{\triangle_{ijk} \supset r} x_{ijk} = 1$ e, do parágrafo anterior temos que, as regiões em R_x cobrem toda a envoltória convexa de P. Assim, r está contida em uma região $r_x \in R_x$ e como todos os triângulos em $sup(x)$ que cobrem r_x também cobre r temos que $\sum_{\triangle_{ijk} \supset r_x} x_{ijk} = 1$. Logo, isto é válido para todos as regiões em R_x, ou $R(P)$ visto que $\bigcup_{r \in R_x} r = \bigcup_{r \in R(P)} r$. □

Com base no Teorema 4.1, o Lema 4.1 também é válido para (Co). Utilizando este fato, podemos finalmente demonstrar o seguinte teorema.

Teorema 4.2 ([dLHSS96]) *Seja P um conjunto finito de pontos no plano. Se todos os pontos de P são vértices de sua envoltória convexa, então toda solução da relaxação linear de (Co) está em coordenadas inteira.*

Prova: Seja x uma solução da relaxação linear de (Co). Seja T' um subconjunto maximal do suporte de x cujos triângulos não se interceptam e cobrem uma única região da envoltória convexa de P. Seja r esta região e $\overline{ij}$ um dos segmentos que a limita. Pela configuração de P não é difícil perceber que r é um conjunto convexo. Logo, pelas equações do cocircuito interno, se $\overline{ij} \notin S_H(P)$, então, supondo que $r \subset L^+_{ij}$, deve existir $\triangle_{ijk} \subset L^-_{ij}$. Como $r \cap \triangle_{ijk} = \emptyset$, então, da maximalidade de T' segue que $\overline{ij}$ deve ser um dos segmentos que limita a envoltória convexa de P. Logo, T' cobre toda a envoltória convexa de P e conseqüentemente é uma triangulação planar de P. Assim, do Lema 4.1 temos que x é uma solução inteira. $\square$

Apesar do resultado acima, no caso geral, o politopo definido pela relaxação linear de (Co) não corresponde à envoltória convexa das soluções inteiras. Considere o conjunto de pontos da Figura 4.4. Observe que o vetor $x \in I\!R^{15}$ com coordenadas $x_{123} = x_{234} = x_{345} = x_{145} = x_{125} = x_{013} = x_{024} = x_{035} = x_{014} = x_{025} = 1/2$ e $x_{012} = x_{023} = x_{034} = x_{045} = x_{015} = 0$. Não é difícil perceber que este vetor satisfaz as equações (III), (IV) e (V). Contudo, não satisfaz a desigualdade de ciclo ímpar $x_{123} + x_{234} + x_{345} + x_{145} + x_{125} \leq 2$, o que mostra que as equações de (Co), ou (PC), não dominam, obrigatoriamente, as desigualdades discutidas na Seção 4.1.

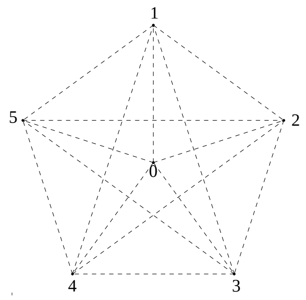


Figura 4.4: Configuração de pontos cuja relaxação linear de (Co) apresenta vértice fracionário.

4.4 O Politopo das Triangulações Minimamente Locais

Todas as desigualdades apresentadas até o momento, seja neste capítulo ou no capitulo anterior, expressam propriedades válidas para todas as triangulação planares. Nesta seção apresentamos um tipo de desigualdade que expressa uma propriedade das triangulações minimamente locais, classe de triangulações planares da qual a TCM faz parte.

Conforme visto na Seção 2.3.2, um segmento $\overline{ij} \in S(P) \backslash S_H(P)$ está em uma triangulação minimamente local de P somente se existem dois triângulos em $\triangle(P)$ que o certifiquem. Logo, um triângulo $\triangle_{k\ell m}$ pode estar em uma triangulação minimamente local de P se cada um de seus lados $\overline{ij} \notin S_H(P)$ também é lado de um outro triângulo $\triangle_{nop} \in \triangle(P)$ de tal forma que $\triangle_{k\ell m}$ e $\triangle_{nop}$ forneçam um certificado para $\overline{ij}$. Esta propriedade pode ser expressa na forma da seguinte desigualdade:

$$x_{k\ell m} - \sum_{\triangle_{nop} \in C(\triangle_{k\ell m}, \overline{ij})} x_{nop} \leq 0 \quad , \forall \triangle_{k\ell m} \in \triangle(P) \text{ e } \overline{ij} \subset \triangle_{k\ell m}. \tag{VI}$$

onde $C(\triangle_{k\ell m}, \overline{ij})$ representa o subconjunto de triângulos de $\triangle(P)$ que juntamente com $\triangle_{k\ell m}$ fornecem um certificado para o segmento $\overline{ij}$, que neste caso é um dos lados de $\triangle_{k\ell m}$.

Note que acresentando as desigualdades (VI), as quais denominamos *desigualdades da minimalidade local*, a (Co) é suficiente para obtemos uma formulação que caracteriza todos os vetores de incidência de triangulações minimamente locais. No próximo capítulo comentamos aspectos observados desta formulação.

Capítulo 5

Resultados Computacionais

Neste capítulo apresentamos os resultados computacionais obtidos utilizando os modelos descritos nos dois últimos capítulos e descrevemos os algoritmos empregados nestes experimentos. O capítulo encontra-se divido em duas seções. Na primeira apresentamos os resultados obtidos com o modelo baseado em segmentos e na segunda os resultados obtidos com o modelo baseado em triângulos.

Todos os algoritmos empregados foram implementados em C++, utilizando o CPLEX 3.0 [CPL94] na resolução das relaxações lineares. No caso especial dos algoritmos desenvolvidos para o modelo baseado em segmentos, utilizamos rotinas e estruturas da biblioteca LEDA [NU] para manipulação de grafos e desenvolvimento de algoritmos de geometria computacional. As instâncias utilizadas foram obtidas determinando-se aleatoriamente (utilizando a rotina `random` do C) a posição de cada ponto dentro de uma grade de dimensões 65535 por 65535. Os tempos de computação apresentados foram obtidos em uma SUN SPARC 1000 com 308Mb de memória e são apresentados em segundos.

5.1 O Modelo Baseado em Segmentos

No Capítulo 3, apresentamos um trabalho cujo principal objetivo era identificar propriedades das triangulações planares que pudessem ser representadas através de PI. Neste sentido apresentamos um conjunto de desigualdades lineares e resultados que são frutos destas propriedades. Nesta seção visamos descrever o trabalho realizado a fim de explorar computacionalmente estas propriedades e resultados. Para isto, tentamos seguir uma linha que possibilite ao leitor entender quais eram as opções disponíveis, qual foi selecionada e por quê.

A primeira parte desta seção é destinada a descrição dos aspectos algorítmicos envolvidos nos experimentos realizados. Começamos por discutir possíveis métodos de pré-processamento, discutimos algumas decisões envolvendo o algoritmo *branch-and-cut* e apresentamos as rotinas de separação empregadas. A segunda parte é dedicada à apresentação dos resultados computacionais e visa basicamente mostrar como combinamos as possibilidades algorítmicas levantadas almejando obter o melhor desempenho computacional possível.

5.1.1 Métodos de Pré-processamento

Costumeiramente, algoritmos para resolução de problemas de PI costumam ser computacionalmente caros. De fato, são métodos elaborados para lidar com problemas "computacionalmente intratáveis", i.e., problemas que não se conhece como resolver em tempo polinomial, em geral, problemas *NP-difíceis*. Diante desta constatação, ganha importância dispormos de bons métodos de pré-processamento, i.e., dispormos de métodos capazes de reduzir o tamanho do problema, de forma a minimizar o trabalho do restante do algoritmo, sem no entanto serem tão dispendiosos a ponto de comprometer o tempo total de processamento.

Como veremos adiante, em nossa abordagem o pré-processamento empregado foi decisivo no desempenho computacional do algoritmo como um todo. Vejamos então quais foram os métodos de pré-processamento testados.

1.17682-esqueleto. Este método consiste tão somente em determinar o 1.17682-esqueleto (Seção 2.3.2) e eliminar do problema todos os segmentos em $S(P)$ que interceptam um dos seus segmentos.

Eliminação por custos reduzidos. Dada uma solução aproximada (viável) para um problema de PI com variáveis que assumem apenas os valores 0 ou 1 (no nosso caso, p.ex., a triangulação gulosa) e uma relaxação linear para este mesmo problema, é possível, dependendo dos custos reduzidos das variáveis e da diferença entre os valores da solução aproximada e da solução da relaxação linear, identificar variáveis que não podem assumir o valor 1 em soluções ótimas [JRT94]. Este método utiliza este resultado com o objetivo de eliminar tais variáveis do problema.

Relaxação linear da formulação do conjunto independente. Este método utiliza o resultado do Teorema 3.1 a fim de eliminar do problema original qualquer segmento cuja variável correspondente assumir o valor 0 na relaxação da formulação de PI cujas únicas restrições são as desigualdades de aresta.

Além dos três métodos mencionados acima, ainda uma outra possibilidade considerada foi a utilização do LMT-esqueleto, de forma análoga ao que foi feito com o 1.17682-esqueleto. Ocorre que, como comentaremos adiante, apesar da clara superioridade do primeiro em identificar subconjuntos de uma TCM, nossa implementação do algoritmo da Figura 2.8 apresentou um tempo de computação excessivamente alto, comprometendo o desempenho do algoritmo como um todo.

5.1.2 Implementação do Branch-and-Cut

No Capítulo 1 apresentamos o algoritmo *branch-and-cut* de forma bastante simples, i.e., apenas um esboço, um pequeno conjunto de passos a serem executados. Entretanto, a experiência mostra que, para obtermos uma implementação eficiente deste tipo de algoritmo muitos outros detalhes devem ser levados em consideração. Em [JRT94], Jünger, Reinelt e Thienel apresentam um *framework* no qual é descrito de forma detalhada quais são estes pontos e como devem ser tratados. Visto que a implementação que realizamos é fortemente baseada neste *framework*, optamos por descrever aqui apenas os principais aspectos que podem diferençar nossa implementação de uma outra baseada no mesmo *framework*, além das rotinas de separação as quais veremos na próxima seção.

Limites Superiores

A fase de *bounding* de um algoritmo *branch-and-bound*, ou *branch-and-cut*, pressupõem que se conhece um limite superior para o custo da solução ótima do problema. Em nossa implementação, na obtenção de tal limite empregamos o custo das triangulações gulosa e de Delaunay. Apesar de não realizarmos aqui uma estudo específico destas duas aproximações da TCM, a triangulação gulosa mostrou-se claramente superior. De fato, para as instâncias nos quais os resultados a serem apresentados foram baseados, a triangulação gulosa sempre apresentou um custo inferior ao da triangulação de Delaunay. Além disso, não encontramos nenhum caso onde o *gap* entre a triangulação gulosa (limite superior) e o limite inferior fornecido pela relaxação linear fosse superior a 12%.

Seleção do Próximo Nó

Detectada a necessidade de se realizar um *branching* a primeira decisão a ser tomada é determinar o nó a ser particionado. Em nossa implementação empregamos uma estratégia tradicional e costumeiramente bem sucedida, escolhemos o nó que apresenta o menor limite inferior. Tradicionalmente denominada de *best bound*, a utilização desta estratégia visou concentrar esforços na tentativa de melhorar mais rapidamente o limite inferior, ao invés do superior. Fazemos isto, pois, no nosso caso, a triangulação gulosa já nos fornece um excelente limite superior.

Regra de Branching

Durante um *branching* o passo seguinte a escolha do nó é a escolha da variável sobre a qual se fará o *branching*. Novamente, aqui empregamos outra estratégia bastante tradicional, escolhemos uma variável cujo valor seja o mais próximo de 0.5 possível. Esta estratégia, além de alterar significativamente a estrutura da solução fracionária, provoca um impacto significativo no custo da relaxação linear ajudando assim a melhorar o limite inferior.

Detecção de Tailing-off

A eficiência de um algoritmo *cutting-planes* está diretamente ligada a velocidade com a qual este converge, i.e., o quão rapidamente o limite inferior fornecido pela relaxação linear, no caso de um problema de minimização, se aproxima do valor da solução ótima. Designamos por *tailing-off* o fenômeno que ocorre com este tipo de algoritmo que faz com que em determinados momentos a velocidade de convergência caia drasticamente. Quando o objetivo primeiro do algoritmo é a eficiência computacional, deve-se pesar quando é melhor realizar um corte ou quando é melhor fazer-se um *branching*. Em um algoritmo do tipo *branch-and-cut*, via de regra, quando detectamos um *tailing-off*, optamos por fazer um *branching*.

No caso da nossa implementação, para detectar o *tailing-off*, verificamos se o limite inferior fornecido pelo valor da relaxação linear não aumentou em pelo menos 0.01 nas últimas 10 iterações.

Heurística Primal

Visando-se melhorar o limite superior, o qual é utilizado na fase de *bounding* para podar nós da árvore de enumeração, podemos utilizar *heurísticas primais*. Tais heurísticas tentam a partir da solução de uma relaxação linear montar uma solução inteira de baixo custo, de preferência com o custo inferior ao do limite superior corrente. Quando isto acontece, o limite superior corrente é atualizado para o valor do custo desta solução.

No nosso caso, tendo em vista que conhecemos um excelente limite superior a priori, não foi implementado nenhuma heurística primal. De fato, a dificuldade em se resolver o problema utilizando esta abordagem mostrou-se estar em provar a otimalidade de uma solução ótima e não em encontrá-la.

Fixação de Variáveis

Como visto na seção de métodos de pré-processamento, para um dado problema de PI é possível através da resolução de sua relaxação linear identificar variáveis que não podem assumir o valor 1 em nenhuma solução ótima. Como em um algoritmo *branch-and-cut* resolvemos uma relaxação a cada iteração, após a resolução de cada uma destas relaxações verificamos a existência de variáveis com a propriedade descrita, se identificamos que uma dada variável x_{ij} possui esta propriedade, então inserimos a equação $x_{ij} = 0$ ao conjunto de restrições corrente.

5.1.3 Rotinas de Separação

Nesta seção descrevemos as rotinas utilizadas pela nossa implentação do algoritmo *branch-and-cut* para separar desigualdades violadas, as rotinas de separação. Tais rotinas são dentro de um algoritmo *branch-and-cut* a parte mais dependente do problema específico que se está tentando solucionar. Em geral, a eficiência do algoritmo depende de forma crucial da eficiência destas rotinas.

Antes, porém, de apresentarmos as rotinas de separação implementadas, deve ficar claro como foram escolhidas as classes de desigualdades que seriam utilizadas como corte no *branch-and-cut*. Tais classes, foram selecionadas basicamente em função de dois parâmetros: sua cardinalidade e a expectativa de eficiência computacional gerada. Ou seja, foram escolhidas classes com um número de desigualdades grande o suficiente para impossibilitar que todas elas fossem inseridas na formulação inicial e que cujas desigualdades se mostraram eficientes em testes iniciais. São elas as desigualdades de clique, ciclo ímpar e planaridade.

Desigualdades de Clique.

Dada uma solução fracionária x' da relaxação linear da formulação de PI válida para o modelo baseado em segmentos, identificar uma desigualdade de clique violada por esta solução equivale ao problema de identificar uma clique de custo máximo no grafo interseção, onde o custo de cada nó representa o valor da variável correspondente. Visto que determinar uma clique de peso máximo em um grafo arbitrário é um problema *NP-difícil* [GJ79], optamos por realizar a separação destas desigualdades de forma heurística. A heurística em questão é bastante simples e foi proposta originalmente por Nemhauser e Sigismondi [NS92].

Seja $ST(v) := \{v\} \cup N(v)$. Por *estrelar* um grafo, entendemos escolher um nó v não marcado, remover do grafo todos os nós que não estão em $ST(v)$ e marcar v. Dado o grafo $G = (V, E)$, escolhendo um nó inicial v, a rotina aplica o processo de estrelamento recursivamente até que todos os nós restantes estejam marcados. Claramente o conjunto de nós não marcados ao final deste processo é uma clique. Na seleção do nó inicial v, a rotina pode utilizar dois critérios distintos:

(a) Escolhe o nó v que maximiza $\{x_v : x_v < 1\}$.

(b) Escolhe o nó v que minimiza $\{|x_v - 1/2| : 0 < x_v < 1\}$.

A motivação para o critério (2) é que a clique de maior peso quando $x_v \geq 0$, para todo $v \in V$, que satisfaz às restrições de aresta é obtida com $x_v = 1/2$ para todo $v \in K$. O critério (1) só é aplicado quando o (2) não consegue determinar uma desigualdade de clique violada.

Desigualdades de Ciclo Ímpar.

Assim como a rotina de separação de desigualdades de clique, a rotina que utilizamos para separar as desigualdades de ciclo ímpar também foi proposta por [NS92]. Porém, neste caso esta rotina é capaz de separar as desigualdades violadas de forma exata.

Considere o grafo $G = (V, E)$. Para cada $(u, v) \in E$, seja $c_{uv} := 1 - x_u - x_v \geq 0$. Inicialmente, construímos um grafo bipartido $G' = (V \cup V', E')$, onde (u, v') e $(v, u') \in E'$ se e somente se $(u, v) \in E$. Claramente, isto pode ser feito em $O(|V| + |E|)$. Seja $c_{uv'} = c_{vu'} = c_{uv}$. Note que um caminho em G' de v_0 a v'_0 é da forma $(v_0, v'_1, v_2, v'_3, ..., v_{2k}, v'_0)$ e o caminho correspondente em G é um ciclo com $2k + 1$ nós. Além disso, o peso do caminho em G' e o peso do ciclo em G é o mesmo e igual a

$$2k + 1 - 2\sum_{j=0}^{2k} x_{v_j}.$$

Conseqüentemente, um caminho de peso mínimo de v_0 a v'_0 produz um ciclo ímpar C em G que maximiza $\sum_{v\in C} x_v - (|C|-1)/2$ dentre todos os ciclos de G contendo v_0, e teremos que $\sum_{v\in C} x_v > (|C|-1)/2$ se e somente se o peso do caminho correspondente no grafo G' for menor que 1. Considere que o algoritmo identificou um ciclo ímpar de peso mínimo C contendo v_0 com $\sum_{v\in C} x_v > (|C|-1)/2$. Podem ocorrer três casos:

(a) $|C| = 3$. Neste caso o ciclo corresponde a uma clique, i.e., identificamos uma desigualdade de ciclo ímpar violada e esta equivale a uma desigualdade de clique.

(b) $|C| \geq 5$ e C é um ciclo ímpar sem cordas. Novamente, temos uma desigualdade de ciclo ímpar encontrada.

(c) $|C| \geq 5$ e C contém uma ou mais cordas. A corda particiona C em um ciclo ímpar C_1 e em um caminho P de tamanho par.

No caso (c) temos

$$\sum_{v\in C_1} x_v + \sum_{v\in P} x_v = \sum_{v\in C} x_v > \frac{|C|-1}{2} = \frac{|C_1|+|P|-1}{2}.$$

Somando-se a restrição de aresta sobre P temos $\sum v \in P x_v \leq |P|/2$. Conseqüentemente

$$\sum_{v\in C_1} x_v > \frac{|C_1|-1}{2}.$$

Agora temos um ciclo ímpar C_1 menor para o qual o argumento acima pode ser repetido. Logo, repetindo-se o procedimento acima sempre podemos obter uma desigualdade de ciclo ímpar violada a partir do ciclo C.

Para todo $v \in V$, a rotina usa o algoritmo de Dijkstra [CLR90] para encontrar o menor caminho entre os nós (v, v') em G'. Se o peso deste caminho é menor que 1, então identificamos uma desigualdade violada.

Desigualdades de planaridade.

Seja x' uma solução fracionária de uma relaxação linear de uma formulação de PI válida para o modelo baseado em segmentos. Uma rotina de separação de desigualdades de planaridade deve tentar identificar conjuntos de pontos $P' \subset P$, tal que $\sum_{\overline{ij} \in S(P')} x_{ij} > 3(|P'|-1) - h'$, onde h' representa o número de pontos em P' sobre a fronteira de sua envoltória convexa. Seja $T'_s \subseteq S(P)$ o conjunto de segmentos tal que as variáveis que os representam, na formulação, assumem valores fracionários em x', i.e., $T'_s = \{\ \overline{ij} \in S(P) : 0 < x'_{ij} < 1\}$. Observe que, se um certo conjunto de pontos $P' \subset P$ não satisfaz a desigualdade de planaridade então, obrigatoriamente, o conjunto de segmentos $S(P') \cap T'_s$ não é vazio, e existe interseção entre seus segmentos. Assim, a idéia da nossa heurística é identificar conjuntos maximais de pontos $P' \subset P$, com $|P'| \geq 4$, tal que para todo ponto em P' exista um segmento em $S(P') \cap T'_s$, do qual ele é um dos extremos, que intercepta pelo menos um outro segmento em $S(P') \cap T'_s$. Para cada um destes conjuntos P', a heurística verifica se a desigualdade de planaridade referente a ele está sendo violada. Em caso afirmativo, esta desigualdade é gerada, i.e, ela é acrescentada à formulação de PI corrente. Caso contrário, remove-se iterativamente pontos de P' até que $|P'| = 3$, ou até que um dos subconjuntos gerados no processo não satisfaça à desigualdade de planaridade. Neste último caso, a desigualdade violada é gerada.

O algoritmo utilizado é apresentado de forma mais detalhada na Figura 5.1, onde $\phi(P') = 3(|P'|-1) - h'$ e h' é o número de pontos em P' sobre a fronteira de sua envoltória convexa. Os passos 1 e 2 do algoritmo utilizam o grafo das interseções de segmentos de P, G_P^{is}, para identificar conjuntos de segmentos de T'_s onde todo segmento intercepta pelo menos um outro segmento do mesmo conjunto. No algoritmo estes conjuntos de segmentos são representados pelos vértices dos componentes conexos em $\mathcal{C}$. Então, no passo 3, para cada um destes conjuntos de segmentos Q, tenta-se determinar se um subconjunto de seus pontos extremos viola a desigualdade de planaridade. Neste sentido, a primeira tarefa realizada é identificar o conjunto $\overline{P}$ de pontos extremos de Q, o que é feito no passo 3(a). Em seguida, no passo 3(b), testa-se se a desigualdade de planaridade referente à $\overline{P}$ está satisfeita. Em caso afirmativo, escolhe-se um ponto $q \in \overline{P}$, faz-se $\overline{P} := \overline{P} \setminus \{q\}$, e repete-se o passo 3(b) para o novo conjunto de pontos $\overline{P}$. Este processo pára se o conjunto de pontos corrente $\overline{P}$ não satisfaz a desigualdade de planaridade, ou se $|\overline{P}| = 3$. No primeiro caso, $|\overline{P}| \geq 4$, e assim o passo 3(c) gera a desigualdade de planaridade correspondente a $\overline{P}$. Observe que, a remoção de pontos de $\overline{P}$, no passo 3(b), é feita de forma gulosa, onde procura-se produzir com a remoção de um ponto $q \in \overline{P}$ o subconjunto de pontos $\overline{P} \setminus \{q\}$ mais próximo à violação de uma desigualdade de planaridade. A complexidade de tempo deste algoritmo é dominada pelo passo 3(b)i, e é igual a $O(|T'_s| n^2 \lg n) = O(|S(P')| n^2 \lg n) = O(n^4 \lg n)$, se utilizarmos o algoritmo *Graham-scan* [CLR90], para computar $\phi(\overline{P})$, no passo 3(b). Ainda assim, na prática, este algoritmo apresenta um bom desempenho computacional. Isto se deve ao fato que, em geral, $|T'_s| \ll |S(P')|$, i.e., o número de variáveis que assumem valores fracionários em uma relaxação linear da formulação da Seção 3.1 é pequeno quando comparado ao total de variáveis desta mesma formulação.

---

1. Determinar o subgrafo $\widehat{G}^{is}_P(\widehat{V}, \widehat{E})$ de $G^{is}_P(V, E)$ induzido por $\widehat{V}$, onde $\widehat{V} = \{v_{ij} \in V : \overline{ij} \in T'_s\}$.
2. Identificar o conjunto $\mathcal{C}$ de componentes conexos de $\widehat{G}^{is}_P$.
3. Para cada componente $\overline{G}^{is}_P(\overline{V}, \overline{E}) \in \mathcal{C}$
   (a) Determinar o conjunto $\overline{P} \subset P$ de todos os pontos que são extremos dos segmentos representados em $\overline{V}$.
   (b) Enquanto $|\overline{P}| \geq 4$ e $\sum_{\overline{ij} \in S(\overline{P})} x_{ij} \leq \phi(\overline{P})$
      i. $pv[p] := \phi(\overline{P} \setminus \{p\}) - \sum_{\overline{ij} \in S(\overline{P} \setminus \{p\})} x_{ij}, \; \forall \, p \in \overline{P}$.
      ii. $q := \min_p\{pv[p] : p \in \overline{P}\}$.
      iii. $\overline{P} := \overline{P} \setminus \{q\}$.
   (c) Se $|\overline{P}| \geq 4$
      Gerar a desigualdade de planaridade referente ao conjunto de pontos $\overline{P}$.

Figura 5.1: Rotina de separação de desigualdades de planaridade.

5.1.4 Resultados Obtidos

Nesta seção apresentamos os resultados dos experimentos computacionais realizados com o modelo baseado em segmentos. Cada linha das tabelas apresentadas traz a média dos valores de cinco instâncias do mesmo tamanho.

Iniciemos por discutir qual a melhor formulação inicial para o algoritmo *branch-and-cut*, i.e., qual a formulação sobre a qual o algoritmo começará a rodar. Na Tabela 5.1 apresentamos uma comparação entre o desempenho de um algoritmo *branch-and-bound* (B&B) cuja formulação inicial é aquela da página 35 e a formulação onde acrescenta-se às restrições da primeira as desigualdades de conjunto dominante. É fácil observar que as desigualdades de maximalidade local melhoram sobremaneira o desempenho computacional do algoritmo.

Vejamos agora, na Tabela 5.2, uma comparação entre o desempenho computacional da segunda formulação mencionada acima, a qual denominamos de *formulação de arestas* e a formulação onde substitui-se as desigualdades de aresta por um conjuntos de desigualdades de clique que recobrem o grafo de interseção de segmentos. Esta última formulação é denominada de *formulação de cliques*. Observe que, apesar da formulação de cliques apresentar um desempenho superior ao da formulação de arestas, o tempo despendido para gerá-la acaba por tornar sua utilização mais dispendiosa. Há de se observar que estes resultados dependem do recobrimento específico que se utiliza. O objetivo primeiro do algoritmo que implementamos para gerar tal recobrimento é minimizar o número total de cliques, minimizando assim o número de desigualdades da formulação. Este algoritmo utiliza a rotina de estrelamento descrita na Seção 5.1.3 para iterativamente determinar um conjunto de cliques que recubra o grafo. A cada iteração uma clique é determinada, seus vértices são removidos do grafo e o processo se repete sobre o grafo restante, sendo que o nó selecionado a cada passo da rotina de estrelamento é aquele que apresenta maior grau dentre todos os vértices do grafo.

Em vista do desempenho apresentado pela formulação de cliques, implementamos um algoritmo capaz de gerar todas as cliques maximais de um grafo, visando determinar na prática a força das desigualdades de clique na resolução do PTCM. Apesar do algoritmo implementado, proposto originalmente por Tsukiyama et. al [TIAS77], possuir tempo de computação linear no número de cliques maximais do grafo, na prática este tempo se mostrou elevado, nos restringindo assim a testes com conjuntos de pontos de cardinalidade reduzida. Tais testes mostraram que as desigualdades de clique são insuficientes para melhorar o desempenho do algoritmo de forma significativa.

A Tabela 5.3, traz uma comparação entre três algoritmos diferentes para resolução do PTCM. Os três algoritmos utilizaram a mesma formulação inicial, a formulação de arestas acrescida das desigualdades do conjunto dominante. Fica claro, que o *branch-and-bound* tem um desempenho significativamente inferior ao *branch-and-cut* (B&C) que utiliza somente os cortes de clique e de ciclo ímpar (sem corte de planaridade), e este por sua vez também apresenta um desempenho muito aquém do apresentado pelo *branch-and-cut* completo (com corte de planaridade).

A Tabela 5.4 apresenta uma comparação dos tempo totais da execução do algoritmo para diferentes métodos de pré-processamento utilizados. Comparamos a utilização do método do 1.17682-esqueleto com a utilização conjugada dos outros dois métodos, aqueles baseados em relaxações lineares. Este último, apesar de facilitar sobremaneira o trabalho do *branch-and-cut* é tão caro que, diferentemente do primeiro, compromete totalmente o tempo total do algoritmo.

Finalmente, nas Tabelas 5.5 e 5.6 apresentamos os melhores resultados obtidos pelo *branch-and-cut*. Os algoritmos utilizados para obter estas duas tabelas diferem apenas pela rotina de separação das desigualdades de planaridade. Os resultados da Tabela 5.5

	Com as desigualdades do conjunto dominante		Sem as desigualdades do conjunto dominante	
n	# nós	Tempo de B&B	# nós	Tempo de B&B
20	3	< 1	633	40
25	7	1	4618	948
30	6	3	8811	8031
35	16	11	36542	18198

Tabela 5.1: Verificação do desempenho computacional da desigualdade do conjunto dominante.

	Formulação de cliques			Formulação de arestas		
n	Tempo gasto para gerar a formulação	Tempo de B&B	Tempo total	Tempo gasto para gerar a formulação	Tempo de B&B	Tempo total
30	6	1	7	< 1	1	1
35	15	5	20	1	13	14
40	32	30	62	1	51	52
45	53	58	111	2	32	34
50	75	175	250	3	243	246

Tabela 5.2: Verificação computacional da formulação que substitui as desigualdades de arestas por uma cobertura de desigualdades de clique.

	B&B		B&C sem planaridade		B&C com planaridade	
n	# nós	Tempo	# nós	Tempo	# nós	Tempo
50	112	168	23	73	1	30
55	230	413	61	171	2	35
60	1497	7684	223	1054	3	71
65	*	*	1544	11475	11	157
70	*	*	*	*	6	214
75	*	*	*	*	12	351
80	*	*	*	*	15	383

*: Média não computada pois a resolução de pelo menos 3 das 5 instâncias em questão ultrapassou o tempo limite imposto de 10 horas.

Tabela 5.3: Comparação entre as diferentes possibilidades de utilização de cortes.

	β-esqueleto			Rel. linear		
n	pré-processamento	B&C	Tempo total	pré-processamento	B&C	Tempo total
50	2	30	32	394	< 1	394
55	4	35	39	643	< 1	643
60	4	71	75	996	1	997
65	7	157	164	1438	2	1440
70	10	214	224	*	*	*
75	11	351	362	*	*	*
80	16	383	399	*	*	*

$*$: Não foi possível realizar o pré-processamento destas instâncias pois a quantidade de memória requerida por este método era superior à quantidade disponível na máquina.

Tabela 5.4: Comparação dos tempos de computação obtidos por duas possibilidades distintas de pré-processamento.

foram obtidos realizando a separação destas desigualdades tal como é descrito no algoritmo da Figura 5.1. Já os resultados apresentados na Tabela 5.6 forma obtidos utilizando uma implementação que substitui os passos 3(b) e 3(c) da da Figura 5.1 por

---

3(b) Se $|\overline{P}| \geq 4$ e $\sum_{\overline{ij} \in S(\overline{P})} x_{ij} \leq \phi(\overline{P})$

Gerar a desigualdade de planaridade referente ao conjunto de pontos $\overline{P}$.

Observe que a o primeiro algoritmo é capaz de resolver o problema com uma árvore de enumeração menor, porém é o segundo quem é mais rápido. Nestas tabelas não consideramos o tempo gasto em pré-processamento na resolução do problema pois este é desprezível (sempre menor que 2 min) quando comparado ao tempo do *branch-and-cut*.

Uma observação interessante com relação aos resultados obtidos, diz respeito a variação do tempo de computação para instâncias do mesmo tamanho. Um exemplo desta variação são as instâncias de 160 pontos (Tabela 5.6). Enquanto uma das instância gastou 5 horas outra gastou apenas 26 minutos. Isto é um indicativo que a dificuldade deste problema, nesta abordagem, está mais intrinsecamente ligada à disposição geométrica dos pontos que ao tamanho da instância.

n	# nós	# PLs	# de cortes de clique	ciclo ímpar	planaridade	Tempo de B&C
100	7	47	0	5	65	1267
110	3	40	1	5	56	1045
120	4	53	1	11	82	1626
130	13	87	0	15	113	2513
140	17	101	0	22	114	8176
150	10	54	0	18	77	2754
160	10	70	0	12	103	3282

Nota: em 16 das 35 execuções do *branch-and-cut* não foi necessário realizar *braching.*

Tabela 5.5: Resultados obtidos utilizando a versão completa da rotina de separação de desigualdades de planaridade ($O(n^6)$).

n	# nós	# PLs	# de cortes de clique	ciclo ímpar	planaridade	Tempo de B&C
100	14	40	5	19	27	940
110	11	43	2	23	39	1022
120	13	39	1	14	37	1090
130	26	69	4	27	50	2125
140	29	71	5	34	42	4064
150	15	54	3	29	47	2706
160	96	146	6	43	48	8359

Nota: em 5 das 35 execuções do *branch-and-cut* não foi necessário realizar *braching.*

Tabela 5.6: Resultados obtidos utilizando a versão da rotina de separação de desigualdades de planaridade que troca o laço do Passo 3(b) por uma simples comparação ($O(n^2)$).

5.2 O Modelo Baseado em Triângulos

Nesta seção descrevemos os resultados computacionais obtidos utilizando o modelo baseado em triângulos, apresentado no Capítulo 4. Naquele capítulo apresentamos três possíveis formulações de PI para o PTCM, sendo que em todas elas as variáveis estavam associadas aos triângulos em $\triangle(P)$. Por motivos descritos naquele capítulo, utilizamos apenas uma destas formulações em nossos testes computacionais: a formulação (Co) (Seção 4.3). Nesta seção discorremos sobre os resultados obtidos com esta formulação e com a formulação na qual acrescentamos às equações de (Co) as desigualdades de minimalidade local (Seção 4.4).

Supreendentemente, para todas instâncias testadas, a relaxação linear de (Co) sempre forneceu uma solução inteira. Contudo, mesmo utilizando uma rotina de pré-processamento, o elevado número de variáveis de (Co) faz com que resolução sua relaxação linear requeira muito espaço. Visando contornar este problema testamos a eficiência da utilização de um algoritmo de geração de colunas.

A primeira parte desta seção é destinada a descrição do algoritmo de geração de colunas empregado. A segunda parte é dedicada a apresentação dos resultados computacionais obtidos com este algoritmo e com a relaxação linear.

5.2.1 Implementação do Algoritmo de Geração de Colunas

Na implementação do algoritmo de geração de colunas utilizamos o ABACUS [Thi96]. O ABACUS consiste em *framework* para implementação de algoritmos de PI tais como *branch-and-bound*, *branch-and-cut* ou *branch-and-price*. Fazendo uso do paradigma de *orientação a objetos*, o ABACUS permite construirmos um destes algoritmos apenas derivando algumas de suas classes (C++) a fim de inserir as rotinas específicas do problema em questão.

A implementação que realizamos é na realidade bastante simplória. Inicialmente, o algoritmo gera todas as variáveis possíveis (triângulos vazios). Parte destas variáveis constituem a base inicial do problema e o restante das variáveis são armazenadas em memória na forma de um *pool* de variáveis. Então, a cada iteração o algoritmo resolve o subproblema corrente e verifica se alguma das variáveis no *pool* apresenta custo reduzido negativo. Em caso positivo, tal variável é inserida no conjunto de variáveis corrente. O algoritmo termina quando não mais existem variáveis com custos reduzidos negativo no *pool*.

Utilizando o ABACUS, o trabalho de implementar o algoritmo acima se reduziu a identificar as variáveis e as restrições a serem utilizadas, determinar quais variáveis iriam compor a base inicial e quais iriam para o *pool*, e determinar quais variáveis do *pool* inserir no subproblema corrente a cada iteração.

Com respeito a base inicial foram testadas duas alternativas: empregamos uma triangulação planar arbitrária e a triangulação gulosa. A utilização da triangulação gulosa se mostrou uma opção nitidamente superior. Com relação a quais variáveis adicionar ao subproblema corrente a cada iteração, a melhor opção, dentre as testadas, foi adicionar todas as variáveis com custo reduzido negativo identificadas.

5.2.2 Resultados Obtidos

Nesta seção apresentamos os resultados obtidos com o modelo baseado em triângulos. Pelos motivos já apresentados, também na resolução do PTCM utilizando este modelo empregamos o método de pré-processamento baseado no 1.17682-esqueleto.

Iniciemos discutindo os resultados obtidos através da simples resolução da relaxação linear de (Co).

Conforme mencionado, para todas instâncias testadas, a relaxação linear de (Co) sempre forneceu uma solução inteira, i.e., um vetor de incidência de uma TCM. Este fato fez com que não houvesse grandes variações nos tempos obtidos para resolver instâncias do mesmo tamanho. Assim, nos testes realizados utilizando este modelo, a dificuldade de se resolver o problema se mostrou mais ligada ao tamanho da instância que à disposição geométrica dos pontos. De fato, o tempo de computação gasto para resolver a relaxação linear de qualquer instância testada foi inferior a um quinto do tempo gasto pela rotina de pré-processamento, a qual possui uma complexidade de tempo igual a $O(n^3)$.

Para ilustrar os fatos acima, apresentamos na Tabela 5.7 os resultados obtidos, utilizando (Co), para instâncias de 500 a 1000 pontos. O tamanho das instâncias representadas nestas linhas é apresentado na coluna 1. A coluna 2 apresenta $|S(P)|$. A coluna 3 apresenta o tempo gasto com o pré-processamento. As colunas 4 e 5 apresentam os números de variáveis e restrições utilizadas nas relaxações lineares, i.e., o número de triângulos e segmentos que não foram eliminados na fase de pré-processamento. A coluna 6 apresenta o tempo de computação gasto para resolver a relaxação linear resultante. E finalmente, a coluna 7 apresenta o tempo total gasto no processo.

Uma primeira observação possível, com relação aos resultados apresentados na tabela 5.7, diz respeito ao desempenho da rotina de pré-processamento. Observe que, para as instâncias testadas, a rotina sempre conseguiu eliminar pelo menos 92% dos segmentos em $S(P)$. Para que se possa ter uma idéia da quantidade de segmentos de uma triangulação

	Pré-processamento		Relaxação linear			
n	# de segmentos	Tempo	# de variáveis	# de restrições	Tempo	Tempo total
500	124750	3746	30766	8975	600	4346
550	150975	4889	36799	10518	751	5640
600	179700	6474	41310	11709	881	7355
650	210925	7767	43261	12371	1032	8799
700	244650	10918	45517	13129	1177	12095
750	280875	12453	50174	14389	1598	14051
800	319600	15081	54095	15487	1880	16961
850	360825	17314	60086	17005	2163	19477
900	404550	23385	63839	18109	2364	25749
950	450775	24233	69089	19419	3139	27372
1000	499500	28881	73679	20626	3725	32606

Tabela 5.7: Resultados computacionais obtidos com a formulação (Co) (Seção 4.3) utilizando somente relaxações lineares.

planar que esta rotina consegue identificar, apresentamos nas Figuras 5.2(a) e 5.2(b), respectivamente, o 1.17682-esqueleto e uma TCM de um mesmo conjunto de 1000 pontos. Outro ponto relevante, com relação aos resultados apresentados na tabela 5.7, é o tempo de computação. Enquanto na abordagem baseada em segmentos foram necessários 8359 segundos (aproximadamente 2,3 horas) para resolver um problema de 160 pontos, utilizando a formulação (Co) fomos capazes de resolver instâncias de 650 pontos em tempo semelhante.

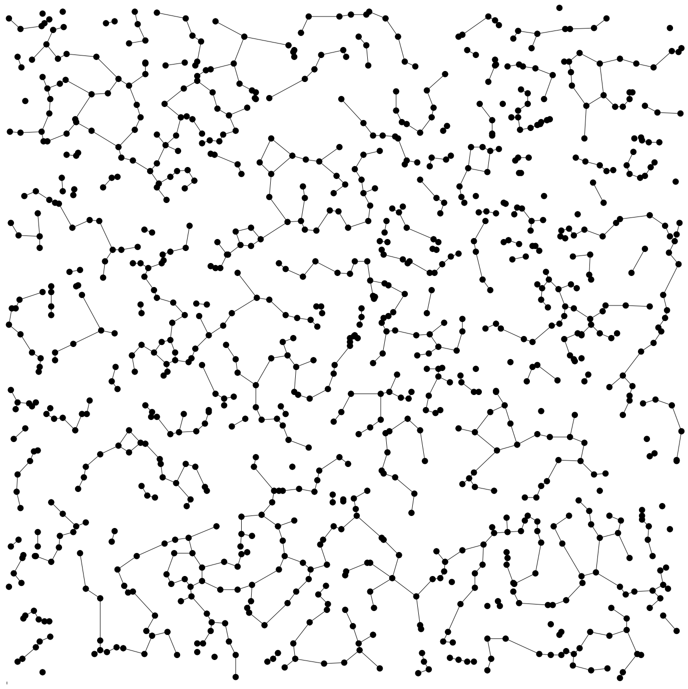
Com respeito ao pré-processamento, através da Figura 5.3, podemos confirmar que, conforme mencionado, O LMT-esqueleto é capaz de determinar um subconjunto de uma TCM significativamente maior que o 1.17682-esqueleto. Contudo, enquanto o tempo gasto para computar este último para um conjunto de 1000 pontos (Figura 5.2(a)) foi aproximadamente 8 horas, o tempo gasto para computar o LMT-esqueleto do mesmo conjunto foi aproximadamente 32 horas.

Ainda através da Figura 5.3, podemos confirmar uma observação feita por Dickerson e Montague [DM96]. Em testes realizados, estes autores constataram que, para conjuntos de pontos gerados aleatóriamente, o LMT-esqueleto sempre produziu um subgrafo conexo de $G = (P, S(P))$. Isto, como visto na Seção 2.3.1, implica que nestes casos o restante dos segmentos que compõem uma TCM podem ser computados em tempo polinomial, e assim o próprio PTCM pode ser resolvido em tempo polinomial. Posteriormente, Bose, Devroye e Evans [BDE96], mostraram que para certa configuração de pontos, denominada *diamond*[1], o LMT-esqueleto não conseguia obter o mesmo resultado. Assim, resolvemos testar a relaxação de (Co) também para esta instância em particular e mais uma vez a relaxação linear forneceu uma solução inteira.

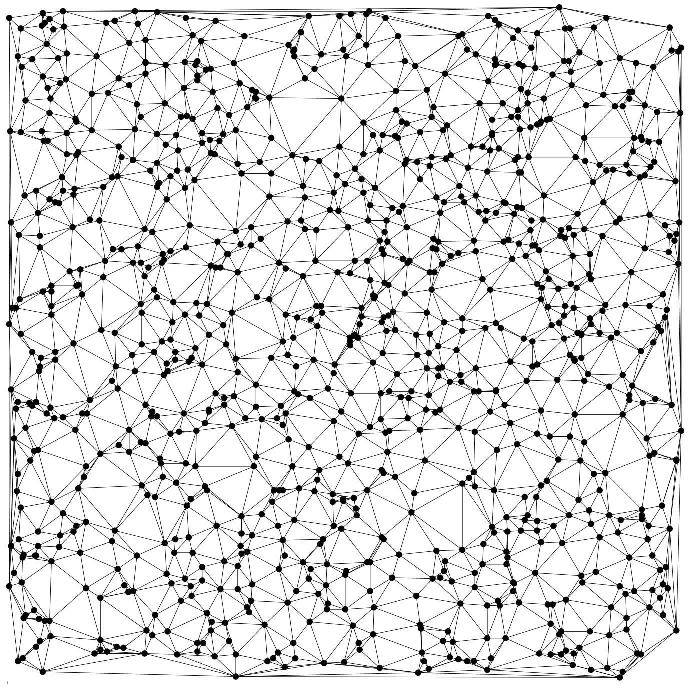
Vale ressaltar novamente que, apesar de que nos testes realizados, utilizando a formulação (Co) para resolver o PTCM, a solução ótima da relaxação foi sempre inteira, o politopo definido por esta formulação não equivale a envoltória convexa das soluções inteiras (veja exemplo na Seção 4.3). Em alguns testes realizados, verificamos que, na maioria dos casos, basta inverter o sentido da função objetivo (maximizar) para que a solução ótima da relaxação linear seja fracionária.

Com relação a esta última observação obtivemos um resultado interessante com as desigualdades de minimalidade local. Gerar tais desigualdades mostrou-se um problema computacionalmente caro, inviabilizando sua utilização computacional. Entretanto, quando adicionamos estas desigualdades as equações de (Co) verificamos, através de testes que, para instâncias de 10 a 40 pontos, que o politopo definido pela relaxação linear coincidiu com a envoltória convexa das soluções inteiras, i.e., das triangulações minimamente locais.

[1]Um *diamond* consiste em um conjunto de 19 pontos onde 18 deles compõem os vértices de um polígono regular com 18 lados e o último ponto está situado no centro do círculo que circunscreve este polígono.



(a) 1.17682-esqueleto.



(b) TCM.

Figura 5.2: Segmentos identificados pelo pré-processamento e a solução final (TCM) de um conjunto de 1000 pontos.

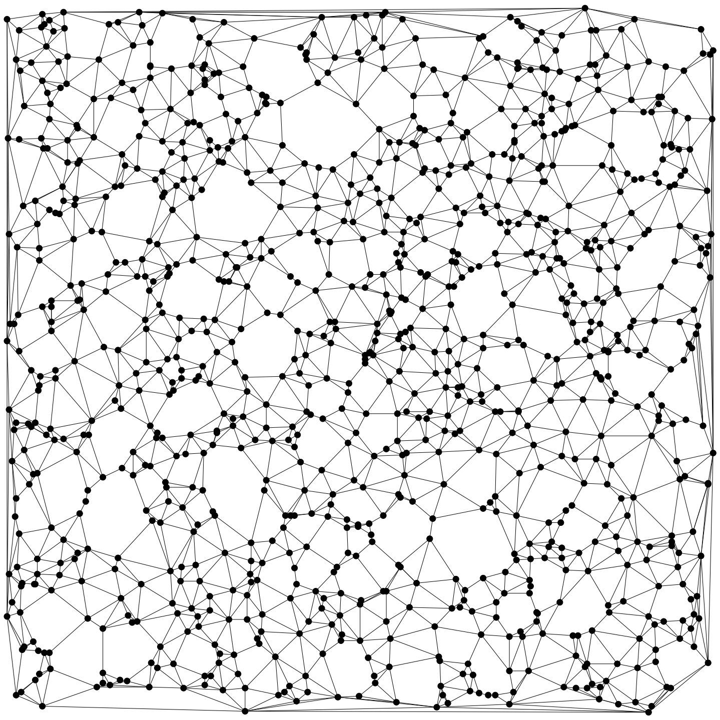


Figura 5.3: LMT-esqueleto de um conjunto de 1000 pontos.

De fato, diferentemente da formulação (Co), não identificamos para esta nova formulação nenhum caso onde o politopo definido por sua relaxação linear coincidiu com a envoltória convexa das soluções inteiras. Para averigarmos este fato, utilizamos o programa LRS [Avi93] para identificar todos os vértices do politopo das referidas relaxações lineares.

Vejamos agora, através da tabela 5.8, o desempenho apresentado pelo algoritmo de geração de colunas. O tamanho das instâncias representadas nestas linhas é apresentado na coluna 1. A coluna 2 apresenta o número de iterações necessárias para resolver o problema. As colunas 3, 4, 5 e 6 trazem, respectivamente, o número total de variáveis do problema, o número de variáveis que compunham o subproblema inicial, o número de variáveis necessárias para resolver o problema e a porcentagem destas com relação ao total. As colunas 7, 8 e 9 apresentam, finalmente, os tempos gastos resolvendo relaxações lineares, identificando as variáveis de custo reduzido negativo no *pool* e o tempo total.

Como pode ser visto, para resolver o problema o algoritmo acaba por utilizar aproximadamente 80% das variáveis, não atendendo assim a um dos objetivos básicos de sua implementação: consumir menos memória. Além disso, o tempo do algoritmo também é muito inferior ao tempo da relaxação linear. Note que o tempo gasto para resolver uma instância de 200 pontos utilizando o algoritmo de geração de colunas é superior ao tempo gasto para resolver uma instância de 700 pontos utilizando uma simples relaxação linear (em ambos os casos com pré-processamento). Desta forma, tal como está implementado não existe vantagem em utilizar o algoritmo de geração de colunas ao invés da relaxação linear, muito pelo contrário. Há de se salientar mais uma vez que nossa implementação do algoritmo de geração de colunas é bastante ingênua. Contudo, os resultados parecem indicar que pequenas modificações não seriam capazes de tonar o algoritmo de geração de colunas mais rápido do que resolver a relaxação linear. Assim, esta última opção parece ser mais adequada desde que, evidentemente, haja memória disponível para carregarmos o problema.

		Variáveis				Tempo		
n	# de LP's	# total	# inicial	# final	% necessária	de LP	de *pricing*	total
100	11	4418	183	3507	79	55	165	220
125	14	5275	232	4121	78	81	279	360
150	17	7466	281	5930	79	433	528	961
175	18	8633	331	6765	78	544	718	1262
200	20	9388	382	7507	80	418	927	1345

Tabela 5.8: Resultados computacionais obtidos com a formulação (Co) (Seção 4.3) utilizando o algoritmo de geração de colunas.

Capítulo 6

Conclusões

Nesta dissertação procuramos apresentar um estudo que possibilitasse avaliar a adequabilidade da utilização de métodos de PI na resolução exata do PTCM. Neste sentido, dois modelos de PI distintos foram avaliados. Em um primeiro instante, nos dedicamos ao estudo das possibilidades e propriedades apresentadas por estes modelos e em seguida utilizamos estes conhecimentos para desenvolver algoritmos capazes de resolver o problema a otimalidade.

Com respeito ao modelo baseado em segmentos, mostramos como formular o PTCM como um problema de PI associando as variáveis deste último aos segmentos em $S(P)$. Mostramos também que existe uma equivalência entre o PTCM e uma versão restrita do problema do conjunto independente em grafos, a qual nos possibilita utilizar resultados já conhecidos para o politopo dos conjuntos independentes na caracterização do politopo das triangulações planares. Além disso, mostramos que é possível fortalecer a formulação proposta através de novas desigualdades válidas que expressam certas propriedades inerentemente geométricas do problema. Finalmente, utilizamos estes resultados para desenvolver um algoritmo *branch-and-cut* para resolver o problema de forma exata.

Utilizando este *branch-and-cut* fomos capazes de resolver instâncias de até 160 pontos em aproximadamente 2 horas. Para obtermos tal performance ficou clara a necessidade de se empregar bons métodos de pré-processamento, no caso o 1.17682-esqueleto. Tão fundamental quanto o método de pré-processamento, foi a utilização das duas novas classes de desigualdades apresentadas, a desigualdade do conjunto dominante e a desigualdade de planaridade. Para esta última, que foi empregada na forma de corte, também foi proposta uma rotina de separação a qual apresentou um excelente desempenho.

Com relação ao modelo baseado em triângulos apresentamos três possíveis formulações de PI para o PTCM. Inicialmente, mostramos que em decorrência da semelhança entre a definição do problema sobre triângulos e sobre segmentos, os principais resultados válidos para o modelo baseado em segmentos, quando devidamente reinterpretados, também são válidos para o modelo baseado em triângulos. Em seguida, baseados no trabalho de Loera et. al [dLHSS96], apresentamos outras duas possibilidades de formulações de PI oferecidas por este modelo. Além disso, apresentamos uma nova classe de desigualdades que é válida somente para o politopo das triangulações minimamente locais, uma classe especial de triangulações planares da qual a TCM faz parte.

Os resultados obtidos com o modelo baseado em triângulos (formulação (Co)), parecem indicar a viabilidade de PI na resolução de exata do PTCM. Com este modelo fomos capazes de determinar a TCM de até 1000 pontos em menos de 10 horas. Como visto, na prática o tempo de computação necessário para resolver uma instância de tamanho n se mostrou da ordem de $O(n^3)$. Entretanto, um problema desta abordagem é a quantidade de memória exigida para tais computações que é da ordem de $O(n^5)$. No sentido de contornar este problema, implementamos um algortimo de geração de colunas, contudo este não apresentou o resultado esperado. Além de não diminuir significativamente a quantidade de memória requerida exige um tempo de computação muito superior a simples resolução de uma relaxação linear.

Para finalizar, dentre os trabalhos encontrados na literatura, os que vêem obtendo maior sucesso na resolução exata do PTCM são aqueles que se utilizam do LMT-esqueleto. De fato, já na apresentação do LMT-esqueleto, Dickerson e Montague [DM96] mostraram que seu algoritmo possibilitava computar, em um Power Macintosh 8500/120, o LMT-esqueleto de 250 pontos em 13 horas. Posteriormente, Belleville, Keil, McAllister e Snoeying mostraram que, utilizando sofisticadas técnicas de geometria computacional, é possível resolver instâncias do PTCM de 1000 pontos, em uma SGI Indy, em aproximadamente 30 minutos. Há de se salientar que, tais trabalhos não se encontravam disponíveis quando do início deste trabalho, época na qual qualquer tentativa de resolver o PTCM de forma exata nos era desconhecida. De qualquer forma, nossa expectativa é de que isto não pode ser feito usando PI, mas talvez se possa otimizar a rotina de pré-processamento, fase que em nossos experimentos determinou o tempo de processamento total, de forma a diminuir a diferença de poder computacional das duas abordagens.

6.1 Possíveis Extensões

Por ter obtido maior sucesso computacional, acreditamos que possíveis extensões deste trabalho poderiam se concentrar no estudo do modelo baseado em triângulos. Do ponto de vista computacional, poder-se-ia tentar otimizar a implementação do método de pré-processamento empregado, i.e., utilizar técnicas de geometria computacional para torná-lo mais rápido, a exemplo do que foi feito com o LMT-esqueleto por Belleville et. al [BKMS96].

Em uma outra linha, poder-se-ia trabalhar no esclarecimento de duas novas questões relacionadas ao PTCM oriundas deste trabalho: a existência de instância com solução ótima fracionária para formulação (Co), quando a função objetivo é a especificada pelo PTCM e existência de instância cujo politopo definido pela relaxação linear de (Co) acrescida das desigualdades de minimalidade local possua vértices fracionários. Com este estudo talvez possa se provar algo sobre a complexidade computacional do PTCM, pelo menos quando a função objetivo é euclideana (ou, no caso mais geral, quando a função objetivo for uma métrica qualquer).

Bibliografia

[AAT+96] O. Aichholzer, F. Aurenhammer, M. Taschwer, S. W. Cheng, N. Katoh, G. Rote, and Y. F. Xu. Triangulations intersect nicely. to appear in Discrete and Computational Geometry, 1996.

[Avi93] D. Avis. A C implementation of the reverse search vertex enumeration algorithm. Technical Report 92.12, School of Computer Science – McGill University, 1993.

[BCCN96] E. Balas, E. Ceria, G. Cornouéjols, and N. Natraj. Gomory cuts revised. *OR Letters*, 19, 1996.

[BDE96] P. Bose, L. Devroye, and W. Evans. Diamonds are not a minimum weight triangulation's best friend. Technical Report TR-96-01, UBC, 1996.

[BEE+93] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T. S. Tan. Edge-insertion for optimal triangulations. *Disc. and Comp. Geometry*, 10:47–65, 1993. Also in *Proc. Latin American Theoretical Informatics*, Springer-Verlag LNCS 583, 1992, pp. 46–60.

[BJ90] M. Bazaraa and J. J. Jarvis. *Linear Programming and Network Flows (2nd ed.)*. John Wiley & Sons, New York, 1990.

[BKMS96] P. Belleville, M. Keil, M. McAllister, and J. Snoeyink. On computing edges that are in all minimum-weight triangulations. Two-page description of video in ACM SCG '96 Video Review, 1996. http://www.cs.ubc.ca/spider/snoeyink/papers.html.

[CC97] E. Cheng and W. H. Cunningham. Wheel inequalities for stable set polytopes. *Mathematical Programming*, 77(3):389–421, 1997.

[Cer95] Sebastián Ceria. Algorithms for integer programming, 1995. Curso apresentado na III Escuela Latinoamericana de Verano en Investigaci'on Operativa.

[CGT95] S. Cheng, M. Golin, and J. Tsang. Expected case analysis of β-skeletons with applications to the construction of minimum-weight triangulations. In *Proceedings of the Seventh Canadian Conference on Computational Geometry*, 1995.

[CLR90] T. H. Cormem, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[CPL94] CPLEX Optimization, Inc. *Using the CPLEX Callable Library*, 1994.

[CX96] Siu-Wing Cheng and Yin-Feng Xu. Approaching the largest β-skeleton within a minimum weight triangulation. Manuscript, 1996.

[DDMW85] M. Dickerson, S. Drysdale, S. McElfresh, and E. Welzl. Fast greedy triangulation algorithms. *Computational Geometry: Theory and Applications*, 1985. to appear.

[DJ89] G. Das and D. Joseph. Which triangulations approximate the complete graph. *Lecture Notes in Computer Science*, 401:168–192, 1989.

[dLHSS96] J. A. de Loera, S. Hosten, F. Santos, and B. Sturmfels. The polytope of all triangulations of a point configuration. 12th European Workshop on Computational Geometry, March 1996.

[DM96] M. T. Dickerson and M. H. Montague. The exact minimum weight triangulation. to appear in Proceedings of the 12th Annual ACM Symposium on Computational Geometry, 1996.

[DRA94] R. L. S. Drysdale, G. Rote, and O. Aichholzer. A simple linear time greedy triangulation algorithm for uniformly distributed points. October 1994.

[dRS94] P. J. de Rezende and J. Stolfi. *Fundamentos de Geometria Computacional*. Recife-PE, 1994. Trabalho apresentado na IX Escola de Computação.

[dS93] C. C. de Souza. *The Graph Equipartition Problem: Optimal Solutions, Extensions and Applications*. PhD thesis, Université Catholique de Louvain, 1993.

[EORW92] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. Finding minimum area k-gons. *Discrete and Computational Geometry*, 7:45–58, 1992.

[Epp94] D. Eppstein. Approximating the minimum weight steiner triangulation. *Discrete and Computational Geometry*, 11:163–191, 1994.

[FW96] C. E. Ferreira and Y. Wakabayashi. *Combinatória Poliédrica e Planos-de-Corte Faciais*. Campinas-SP, 1996. Trabalho apresentado na X Escola de Computação.

[Gil79] P. D. Gilbert. *New Results on Planar Triangulations*. PhD thesis, University of Illinois, 1979. Report No. UILUENG 78 2243.

[GJ79] M. R. Garey and D. S. Johnson. *Computer and intractability: A guide to the theory of NP–completeness*. Freeman, San Francisco, 1979.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.

[GP85] M. Grötschel and M. W. Padberg. Polyhedral theory. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem*, chapter 8, pages 251–306. 1985.

[GS85] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[HP94] L. S. Heath and S. V. Pemmaraju. New results for the mimimum weight triangulation problem. *Algorithmica*, 12:533–552, 1994.

[HS84] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press, 1984.

[JNS97] E. L. Johnson, G. L. Nemhauser, and M. W. P. Savelsbergh. Progress in integer programming: An exposition. Submitted to INFORMS Journal on Computing, 1997.

[JRT94] M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. Technical Report 94.156, Universität zu Köln, Germany, 1994.

[Kei94] J. M. Keil. Computing a subgraph of the minimum weight triangulation. *Computational Geometry: Theory and Applications*, 4:13–26, 1994.

[Kir80] D. G. Kirkpatrick. A note on delaunay optimal triangulations. *Information Processing Letters*, 10(3):127–128, 1980.

[KR85] D. G. Kirkpatrick and J. D. Radke. A framework for computational morphology. In *Computational Geometry, 1985, North-Holland*. 1985.

[Kyo96] Y. Kyoda. A study of generating minimum weight triangulation within practical time. Master's thesis, Department of Information Science, Univesity of Tokyo, 1996.

[Lev87] C. Levcopoulos. An $\Omega(\sqrt{n})$ lower bound for the greedy triangulation. *Information Processing Letters*, 25(4):247–251, 1987.

[LK94] C. Levcopoulos and D. Krznaric. The greedy triangulation can be computed from the Delaunay in linear time. Technical Report LU-CS-TR:94-136, Departament of Computer Science, Lund University, Lund, Sweden, 1994.

[LK96a] C. Levcopoulos and D. Krznaric. Quasi-greedy triangulations approximating the minimum weight triangulation. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401, Atlanta, Georgia, 1996.

[LK96b] C. Levcopoulos and D. Krznaric. Tight lower bounds for minimum weight triangulation heristics. *Information Processing Letters*, 57:129–135, 1996.

[LL87] C. Levcopoulos and A. Lingas. On approximation behavior of greedy triangulation for convex polygons. *Algorithmica*, 2:175–193, 1987.

[LL91] C. Levcopoulos and A. Lingas. Greedy triangulation approximates the optimum and can be implemented in linear time in the average case. *Lecture Notes in Computer Science*, 497:139–148, 1991.

[Llo77] E. L. Lloyd. On triangulations of a set of points in the plane. In *Proc. 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 228–240, 1977.

[Min86] M. Minoux. *Mathematical Programming: Theory and Algorithms*. Wiley-Interscience, 1986.

[MZ79] G. K. Manacher and A. L. Zobrist. Neither the greedy nor the delaunay triangulation of a planar point set approximates the optimal triangulation. *Information Processing Letters*, 9(1):31–34, 1979.

[NdS96] A. P. Nunes and C. C. de Souza. Uma abordagem de programação inteira para o problema da triangulação de custo mínimo. Technical Report IC–96-19, Instituto de Computação – Unicamp, 1996.

[NS92] G. L. Nemhauser and G. Sigismondi. A strong cutting plane/branch-and-bound algorithm for node packing. *Journal of Operational Research Society*, 43(5):443–457, 1992.

[NT74] G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1974.

[NT75] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

[NU] S. Näher and C. Uhrig. *The LEDA User Manual Version R 3.3.c.*

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, New York, 1988.

[Pad73] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[PH87] D. A. Plaisted and J. Hong. A heuristic triangulation algorithm. *Journal of Algorithms*, 8(3):405–437, 1987.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[Sch86] A. Schrijver. *Theory of Linear and Interger Programming.* John Wiley and Sons, Chichester, 1986.

[Tan93] T. S. Tan. *Optimal Two-Dimesional Triangulations.* PhD thesis, Univerity of Illinois at Urbana-Champaign, 1993.

[Thi96] S. Thienel. *ABACUS - A Branch-And-CUt System, Version 1.2, User's Guide and Reference Manual*, 1996.

[TIAS77] Tsukiyama, Ide, Ariyoshi, and Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 1977.

[Tro75] L. E. Trotter. A class of facet producing graphs for vertex packing poliedra. *Discrete Mathematics*, 12:373–388, 1975.

[WA86] C. A. Wang and J. K. Aggarwal. Surface reconstruction and representation of 3d-scenes. *Pattern Recognition*, 19:223–232, 1986.

[Wol76] L. A. Wolsey. Further facet generating procedures for vertex packing polytopes. *Mathematical Programming*, 11:158–163, 1976. Short Communication.

[Yan95] B. T. Yang. A better subgraph of the minimum weight triangulation. *Information Processing Letters*, 56:255–258, 1995.

[Yoe75] P. Yoeli. Compilation of data for computer-assisted relief cartography. In J. C. Davis and M. J. McCullagh, editors, *Display and Analysis of Spatial Data*, pages 352–367. Wiley, New York, 1975.

[YXY94] B.-T. Yang, Y.-F. Xu, and Z.-Y. You. A chain decomposition algorithm for the proof of a property on minimum weight triangulations. *Lecture Notes in Computer Science*, 834:423–427, 1994.