Algoritmos de Aproximação para o Problema de Classificação Métrica

Evandro Cesar Bracht

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Algoritmos de Aproximação para o Problema de Classificação Métrica

Evandro Cesar Bracht[1]

maio de 2004

Banca Examinadora:

- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação, Unicamp (Orientador)
- Prof. Dr. Orlando Lee
  Instituto de Computação, Unicamp
- Prof. Dra. Cristina Gomes Fernandes
  Instituto de Matemática e Estatística, USP
- Prof. Dr. Ricardo Dahab
  Instituto de Computação, Unicamp (Suplente)

[1] **Auxílio financeiro do FAPESP**

Algoritmos de Aproximação para o Problema de Classificação Métrica

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Evandro Cesar Bracht e aprovada pela Banca Examinadora.

Campinas, junho de 2004.

Prof. Dr. Flávio Keidi Miyazawa
Instituto de Computação, Unicamp (Orientador)

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

Nunca desista. Coloque no Easy.
(Gregorio Bagio Tramontina)

Resumo

Em um problema de classificação tradicional temos um conjunto de n objetos e um conjunto de m classes e queremos classificar cada objeto como pertencente a uma classe, de modo que esta classificação seja consistente com alguns dados que temos sobre o problema. Este trabalho apresenta um estudo do problema de classificação métrica através de algoritmos aproximados. Os algoritmos aproximados conhecidos para este problema são baseados na solução de grandes programas lineares e são impraticáveis para instâncias de tamanho moderado e grande. Apresentamos um algoritmo $8 \log n$-aproximado, analisado pela técnica primal-dual, que apesar de possuir fator de aproximação maior que os algoritmos anteriores, pode ser aplicado a grandes instâncias. Mostramos também que este fator de aproximação é justo, exceto por um fator constante. Obtivemos resultados experimentais usando instâncias geradas computacionalmente e instâncias de processamento de imagens com o novo algoritmo e com outros dois algoritmos baseados na resolução de programas lineares. Para estas instâncias o algoritmo proposto apresentou soluções de boa qualidade com um ganho considerável no tempo computacional.

Abstract

In a traditional classification problem, we have a set of n objects and a set of m labels (or classes). We wish to assign one of m labels (or classes) to each one of n objects, in a way that is consistent with some observed data that we have about the problem. In this work we present a study of approximation algorithms for the metric labeling problem. The known approximation algorithms for this problem are based on solutions of large linear programs and are impractical for moderate and large size instances. We present an $8 \log n$-approximation algorithm analyzed by a primal-dual technique which, although has a factor greater than the previous algorithms, can be applied to large sized instances. We also show that the analysis is tight, up to a constant factor. We obtained experimental results on computational generated and image processing instances with the new algorithm and two other LP-based approximation algorithms. For these instances our algorithm presents good quality solutions with a considerable gain of computational time.

Agradecimentos

Primeiramente a Deus, por não deixar faltar força e vontade para que este trabalho se concluísse.

Ao professor orientador, Dr. Flávio Keidi Miyazawa, pela convivência sincera e amiga, apoio constante em todos os momentos, trabalho exímio, paciência e bom humor constante.

Aos meus amigos pelo companheirismo, apoio e compreenção. A alguns amigos em especial, que estiveram mais próximos durante este tempo, auxiliando no desenvolvimento do trabalho ou simplesmente companheiros em todos os momentos: Alexandre (Capitão), Gregório, Fernando, Giovani, Luis Augusto, Eduardo (Pará), Silvana, Borin, Juliana... e por aí vai.

A minha família, especialmente a minha mãe Metildes pelo apoio às minhas decisões e por sempre acreditar em mim.

À FAPESP pelo suporte financeiro.

A todos aqueles que não impediram a conclusão desse trabalho.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Quando pensamos em classificação a primeira idéia que surge é que temos um conjunto de objetos e queremos atribuir cada objeto a uma classe de modo que objetos iguais ou muito similares pertençam a mesma classe. Na área de computação existem muitos problemas que possuem uma definição similar, como por exemplo, os problemas de particionamento (*clustering*) ou o problema de localização de recursos. No entanto, nenhum destes problemas possui as mesmas características do problema de Classificação Métrica.

No problema de classificação métrica temos um conjunto de n objetos e um conjunto de m classes e queremos classificar cada objeto como pertencente a uma classe. A escolha da melhor classificação para os objetos leva em consideração dois conjuntos de custos, o custo de atribuição e o custo de separação, que são descritos a seguir.

- **Custo de atribuição:** é o valor pago por associar um objeto a uma classe. Este valor é pago apenas uma vez para cada objeto, porque um objeto só pode pertencer a uma classe.

- **Custo de separação:** No problema de classificação métrica temos informações sobre a relação entre pares de objetos. Quanto mais forte for a relação entre dois objetos, maior é a sua similaridade. Temos também informações sobre a similaridade entre as classes. O custo de separação é pago quando dois objetos, relacionados, são associados a classes diferentes, sendo que seu valor é proporcional à força da relação existente entre os objetos e à dissimilaridade existente entre as classes.

Esta abordagem para o problema de classificação é aplicável a várias áreas, tais como, processamento de imagens, análise biométrica, visão computacional, etc.

Uma das aplicações na área de processamento de imagens é o problema de restauração de imagens degeneradas por ruídos, onde os pontos da imagem são considerados objetos e as cores classes. O custo de associação de uma nova cor a um ponto é definido de acordo com a similaridade entre a nova cor e a cor atual do ponto. Por exemplo, se a cor atual do ponto

for 255 e a nova cor for 55, o custo de se associar esta nova cor ao ponto é, digamos, de 200. A relação entre dois pontos, que é um dos fatores do custo de separação, é definida através da coloração atual destes pontos e a distância entre eles na malha de pontos. Assim se u e v são dois pontos vizinhos na malha de pontos, e a cor de u é igual a cor de v, então a força da relação entre eles é um valor x muito alto. Se associarmos os objetos u e v a cores diferentes, o custo de separação para estes objetos é x vezes a distância entre as cores associadas.

Uma outra aplicação, também em processamento de imagens, é a identificação de regiões em uma imagem, onde cada ponto da imagem é visto como um objeto e cada região pode ser vista como uma classe, assim pontos classificados na mesma classe pertencem à mesma região.

Outro exemplo de aplicação do problema de classificação métrica é o problema de classificação de hipertextos, onde os documentos são os objetos que desejamos classificar e as classes são as categorias. A relação entre os objetos é definida através da existência de *links* entre os documentos e da ocorrência de palavras chaves em cada documento. O custo de associação de um objeto a uma classe é estimado com base no número de palavras chaves que o documento possui relacionadas à categoria que a classe representa.

Nesta dissertação estudamos o problema de classificação métrica do ponto de vista de algoritmos aproximados. Faremos uma breve revisão de alguns conceitos da área de algoritmos aproximados e apresentaremos o problema de classificação métrica e suas variações, juntamente com algumas abordagens encontradas na literatura. Apresentaremos também, um novo algoritmo guloso para o problema de classificação uniforme, que surgiu do estudo de problemas relacionados, como o problema de localização de recursos e o problema de cobertura por conjuntos.

Este novo algoritmo, apesar de possuir um fator de aproximação maior que o fator de aproximação de algoritmos já existentes, tem tempo de execução prático menor e conseqüentemente pode ser aplicado a instâncias maiores. Isto ocorre porque a grande maioria dos algoritmos encontrados na literatura, para o problema de classificação métrica, são baseados na resolução de programas lineares grandes.

Apresentaremos também os resultados experimentais comparando tempos de execução e qualidade das soluções obtidas para o algoritmo proposto e outros baseados em programação linear.

1.1 Organização da Dissertação

Inicialmente, no capítulo 2, apresentaremos alguns conceitos utilizados na abordagem de algoritmos aproximados, passando a seguir para a descrição formal do problema de classificação métrica e suas variantes, bem como apresentação de problemas correlatos.

Nos capítulos 3 e 4 são discutidos os principais algoritmos encontrados na literatura para o problema de classificação métrica, que estão divididos em algoritmos que utilizam programação

linear, no capítulo 3, e no capítulo 4 algoritmos que utilizam técnicas de fluxo em rede.

Além disso, no capítulo 5, apresentamos um novo algoritmo para o problema de classificação métrica e o seu fator de aproximação, juntamente com alguns problemas que serviram de inspiração para a construção deste algoritmo. Neste capítulo mostraremos também que o fator de aproximação é justo, exceto por um fator constante.

Por fim, no capítulo 6 é mostrado uma comparação do desempenho do novo algoritmo com o desempenho dos algoritmos apresentados anteriormente.

Capítulo 2

Preliminares

2.1 Algoritmos Aproximados

Nesta seção apresentamos uma visão geral sobre algoritmos de aproximação. Apresentamos uma notação comum na área de algoritmos de aproximação bem como definições e conceitos básicos. Além disso, discutimos brevemente quais técnicas são utilizadas para se lidar com problemas de aproximação.

A motivação de se desenvolver algoritmos de aproximação é a de obter algoritmos rápidos (polinomiais), que produzam soluções dentro de um fator de aproximação.

Nos últimos anos, surgiram várias técnicas de caráter geral para o desenvolvimento de algoritmos de aproximação. Algumas destas são: *arredondamento de soluções via programação linear, dualidade em programação linear e método primal-dual, algoritmos probabilísticos e sua desaleatorização, programação semidefinida,* dentre outras (veja [5, 12, 22, 23]).

Uma estratégia comum para se tratar problemas de otimização combinatória é formular o problema através de um programa linear inteiro e resolver a relaxação linear deste, uma vez que isto pode ser feito em tempo polinomial. A solução obtida através do programa linear pode ser fracionária e muitas vezes não é aplicável ao problema. Uma abordagem muito comum nestes casos é o arredondamento das soluções fracionárias geradas pelo programa linear.

Outra técnica, que envolve a formulação do problema como um programa linear, é resolver o sistema dual do programa linear, em vez do primal, e em seguida obter uma solução com base nas variáveis duais. Uma outra técnica mais recente é o uso do método de aproximação primal-dual, que tem sido usado para obter diversos algoritmos combinatórios usando a teoria de dualidade em programação linear. Neste caso, o método é em geral combinatório, não requerendo a resolução de programas lineares e consiste de uma generalização do método primal-dual tradicional.

Já no caso de algoritmos probabilísticos, o algoritmo contém passos que dependem de uma seqüência de bits aleatórios, dados na entrada. Neste caso temos um valor esperado para a

solução gerada pelo algoritmo que é calculado com base nas probabilidades de todas as escolhas aleatórias. É interessante observar que apesar do modelo parecer restrito, a maioria dos algoritmos probabilísticos podem ser desaleatorizada, através do método das esperanças condicionais, tornando-os algoritmos determinísticos (veja [5]). A versão probabilística é em geral mais simples de se implementar e mais fácil de se analisar que a correspondente versão determinística. Além disso, muitos dos algoritmos de aproximação combinam o uso de técnicas de programação linear com técnicas usadas em algoritmos probabilísticos, considerando o valor das variáveis obtidas pela relaxação linear como probabilidades. Estas técnicas, tanto isoladamente como em conjunto, têm sido aplicadas com sucesso em diversos problemas nos últimos anos.

Cada algoritmo de aproximação possui um fator de aproximação, que nos diz o quão longe pode estar a solução obtida de uma solução ótima. Podemos definir o fator de aproximação para um algoritmo da seguinte forma.

Definição 2.1.1 *Dado um algoritmo $\mathcal{A}$ para um problema de otimização, se I é uma instância para este problema, denotamos por $\mathcal{A}(I)$ o valor da solução devolvida pelo algoritmo $\mathcal{A}$ aplicado à instância I e denotamos por $\mathrm{OPT}(I)$ o correspondente valor para uma solução ótima de I. Para problemas de minimização (maximização), diremos que um algoritmo $\mathcal{A}$ tem um fator de aproximação α, ou é α-aproximado, se $\mathcal{A}(I)/\mathrm{OPT}(I) \leq \alpha$, com $\alpha > 1$ ($\mathcal{A}(I)/\mathrm{OPT}(I) \geq \alpha$, com $\alpha < 1$), para toda instância I. No caso dos algoritmos probabilísticos, consideramos a desigualdade $E[\mathcal{A}(I)]/\mathrm{OPT}(I) \leq \alpha$ ($E[\mathcal{A}(I)]/\mathrm{OPT}(I) \geq \alpha$), onde a esperança $E[\mathcal{A}(I)]$ é tomada sobre todas as escolhas aleatórias feitas pelo algoritmo.*

Para mostrar que um algoritmo é de aproximação, o primeiro passo é buscar uma prova de seu fator de aproximação. Posteriormente é interessante encontrar alguma instância para o algoritmo cujo valor da solução produzida pelo algoritmo atinja o fator de aproximação em relação ao valor ótimo. Neste caso dizemos que o fator de aproximação do algoritmo é justo. Podemos definir um fator de aproximação justo da seguinte forma.

Definição 2.1.2 *Dado um algoritmo $\mathcal{A}$, para um problema de minimização (maximização), com fator de aproximação α, dizemos que α é um fator de aproximação justo se existir, para qualquer $\epsilon \geq 0$, pelo menos uma instância I, tal que $\mathcal{A}(I)/\mathrm{OPT}(I) \leq \alpha - \epsilon$ ($\mathcal{A}(I)/\mathrm{OPT}(I) \geq \alpha + \epsilon$).*

Do ponto de vista teórico os algoritmos de aproximação mais desejados são aqueles que tem fator de aproximação próximo de 1, embora isto nem sempre seja possível.

2.2 O problema de Classificação Métrica

Primeiro, vamos estabelecer algumas convenções básicas de notação. Dada uma função $c : E \times E \to \mathbb{R}$ que associa um número a cada par de elementos $\{x, y\}$ de um conjunto finito E, denotamos o valor de c em $\{x, y\}$ por c_{xy}. Dizemos que uma função c é métrica somente se esta função respeitar a desigualdade triangular, ou seja, dado três elementos x, y e z quaisquer, $c_{xy} \leq c_{xz} + c_{zy}$. Se uma função c respeitar a desigualdade triangular então, c é métrica.

Podemos definir uma classificação da seguinte forma:

Problema 2.2.1 *Classificação é uma função de classificação $f : P \to L$, onde P é um conjunto de n objetos e L é um conjunto de m classes.*

No problema de classificação métrica temos um conjunto P de n objetos e um conjunto L de m classes e o objetivo é associar cada objeto de P a alguma classe em L, de modo que o custo dessa classificação seja minimizado. Para definir o custo de uma classificação levamos em consideração as seguintes informações:

- **Custo de atribuição**, definido pela função $c : P \times L \to \mathbb{Q}^+$. Quando um objeto $u \in P$ é associado a uma classe $i \in L$ através de uma função de classificação f, isto é, $f(u) = i$, o custo $c(u, i)$ é pago e denotamos por $A(u, f)$ este valor. O custo de atribuição é pago apenas uma vez para cada objeto, porque um objeto só pode pertencer a uma classe. O custo de atribuição total de uma classificação f, para todos os objetos em P, é $A(P, f) = \sum_{u \in P} A(u, f)$.

- **Distância métrica** entre as classes, definida pela função $d : L \times L \to \mathbb{Q}^+$, indica a dissimilaridade entre as classes, ou seja, quanto maior o valor de $d(i, j)$ para $i, j \in L$ menor é a similaridade entre estas classes.

- **Força da relação** entre dois objetos, definida pela função $w : P \times P \to \mathbb{Q}^+$, que indica qual a similaridade entre dois objetos, ou seja, quanto maior é o valor de $w(u, v)$ para $u, v \in P$, maior é a similaridade entre estes objetos.

- **Custo de separação** é o valor pago por associar dois objetos u e v à classes diferentes. Este valor é proporcional à dissimilaridade existente entre as classes e à força da relação entre os objetos. Se o objeto $u \in P$ for associado à classe $i \in L$ e o objeto $v \in P$ for associado à classe $j \in L$ através de uma função de classificação f, o custo de separação pago para os objetos u e v é $w(u, v) \cdot d(i, j)$. Denotamos este valor por $S(u, v, f)$. O custo de separação para todos os objetos em P para uma classificação f é $S(P, f) = \sum_{u,v \in P} S(u, v, f)$.

Podemos definir o problema de classificação métrica da seguinte forma:

Problema 2.2.2 *Uma instância para o problema de classificação métrica (Metric Labeling Problem (ML)) é uma tupla* (P, L, d, c, w) *onde*

- P *é um conjunto de* n *objetos,*

- L *é um conjunto de* m *classes,*

- $d : L \times L \to \mathbb{Q}^+$ *é uma função de distância métrica,*

- $c : P \times L \to \mathbb{Q}^+$ *é o custo de associação,*

- $w : P \times P \to \mathbb{Q}^+$ *é a força da relação entre objetos,*

e o objetivo é encontrar uma função de classificação $f : P \to L$*, tal que,* $A(u, f) = c(u, f(u))$ *e* $S(u, v, f) = w(u, v) \cdot d(f(u), f(v))$*, e que minimize o custo*

$$Q(f) = A(P, f) + S(P, f) = \sum_{u \in P} c(u, f(u)) + \sum_{u,v \in P} w(u, v)d(f(u), f(v)).$$

Note que podemos interpretar o conjunto de objetos P e o peso w da força da relação entre objetos como um grafo $G = (P, E)$ com pesos nas arestas, onde o conjunto dos vértices do grafo é composto pelos objetos do conjunto P e a relação entre dois objetos $u, v \in P$ é representada por uma aresta não orientada $e = \{u, v\} \in E$, com peso $w(u, v)$. Daqui em diante podemos referenciar o peso $w(u, v)$ de uma aresta $e = \{u, v\}$ como w_e.

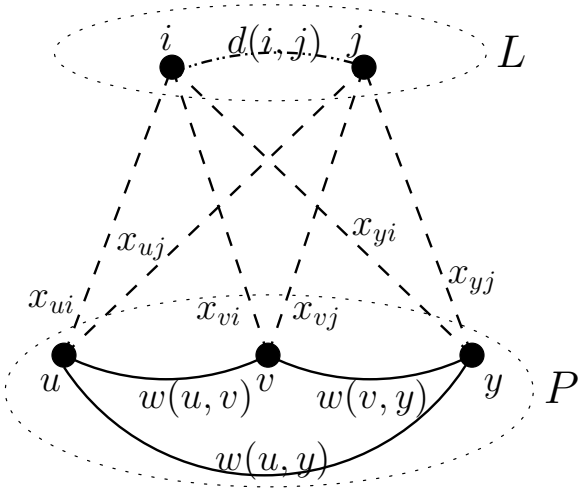

Figura 2.1: Exemplo de uma instância para o problema de classificação métrica.

A figura 2.1 representa uma instância para o problema de classificação métrica e uma possível classificação seria $f(u) = i, \quad f(v) = j, \quad f(y) = i$. Esta classificação teria os seguintes custos:

- custo de classificação: $c(u, f(u)) + c(v, f(v)) + c(y, f(y)) = c_{ui} + c_{vj} + c_{yi}$;
- custo de separação: $w(u, v) + w(y, v)$.

Assim o custo de classificação final seria $c_{ui} + c_{vj} + c_{yi} + w(u, v) + w(y, v)$.

Esta abordagem do problema de classificação é aplicável a várias áreas, tais como, processamento de imagens, análise biométrica, visão computacional, etc.

Um exemplo de aplicação do problema de classificação métrica é a recuperação de imagens degeneradas por ruídos. Nesta aplicação uma imagem é vista como uma malha de pontos (*grid*) onde cada ponto possui uma cor, que pode ser sua cor "real" ou o resultando da degeneração da imagem por ruídos. Para o problema de classificação, os pontos são os objetos e as cores são as classes. O objetivo é encontrar a "verdadeira" coloração dos pontos, ou seja, a nova classificação, que é a imagem sem ruídos.

O custo de associação de um objeto a uma classe é estimado através da coloração atual do objeto e a cor que a classe representa, e através da posição de dois objetos na malha de pontos e suas cores podemos obter a relação entre estes dois objetos.

O objetivo é encontrar a melhor classificação para o conjunto de pontos, onde a "melhor classificação" é baseada em duas influências competitivas: o custo de associar uma cor a um ponto, e o "castigo" pago por associar dois pontos relacionados, pertencentes a uma mesma região da imagem, a cores diferentes. A melhor recuperação da imagem é obtida quando o somatório dos custos de associação com os custos de separação for minimizada.

O problema de classificação métrica possui alguns casos especiais. A principal diferença entre estes casos está na função de distância aplicada ao conjunto das classes. Podemos citar os seguintes casos especiais:

- **Problema de Classificação Métrica Linear (*Linear Metric Labeling Problem (LML)*):** Possui as mesmas características que o problema ML, com a diferença que cada classe é dada por um número inteiro positivo não repetido e a função de distância entre as classes é dada pelo módulo da diferença entre seus números, i.e., $d : L \times L \rightarrow \mathbb{N}^+$ é tal que $d_{ij} = |i - j|$.

- **Problema de Classificação Métrica Linear Truncada (*Truncated Linear Metric Labeling Problem (TML)*):** Possui as mesmas características que o problema ML, com a diferença que cada classe é dada por um número inteiro positivo não repetido e a função de distância entre as classes é dada pelo menor valor entre o módulo da diferença entre as duas classes e um valor máximo M, i.e., $d : L \times L \rightarrow \mathbb{N}^+$ satisfaz $d_{ij} = \min\{|i - j|, M\}$.

- **Problema de Classificação Uniforme (*Uniform Metric Labeling (UML)*):** Possui as mesmas características que o problema ML, com a diferença que a função de distância entre as classes sempre devolve 1 para quaisquer duas classes, i.e., $d_{ij} = 1 \ \ \forall i, j \in L$ com $i \neq j$ e 0 caso contrário.

Boykov, Veksler e Zabih [4] desenvolveram a conexão entre o problema de classificação métrica uniforme, com $m > 2$, e o problema de multi-cortes em grafos (*multiway cuts*) e

mostraram uma redução direta do problema de multi-cortes para o problema de classificação métrica uniforme. Assim, como o problema de multi-cortes é um problema NP-difícil, o problema de classificação métrica uniforme também é.

2.3 Problemas Correlatos

Existem problemas que são muito parecidos com o problema de classificação métrica. Alguns destes são os problemas de localização de recursos, particionamento, cobertura por conjuntos, entre outros. Devido à grande similaridade existente entre estes problemas e o problema de classificação métrica, muitas das idéias desenvolvidas para estes problemas correlatos podem ser úteis para o nosso problema. A seguir descrevemos um pouco mais sobre alguns destes problemas.

2.3.1 Problema de Localização de Recursos

Desde o início dos anos 60 o problema de localização de recursos *(Facility Location Problem)* ocupa um ponto central na área de pesquisa operacional. Este problema modela situações de definição do local de instalação de indústrias, armazéns, escolas, e hospitais e mais recentemente incluem a instalação de servidores de proxy na web [22], instalação de centrais telefônicas, entre outros.

Problema 2.3.1 *No Problema de Localização de Recursos temos um conjunto $F = \{1, \ldots, n\}$ de recursos, e um conjunto $C = \{1, \ldots, m\}$ de clientes, que devem ser atendidos pelos recursos. Temos também um custo $c_{ij} \in \mathbb{Z}$, onde $1 \leq i \leq n$ e $1 \leq j \leq m$, que representa o custo de se atender o cliente j pelo recurso i. Se algum cliente for associado ao recurso i então temos que pagar um custo f_i para abrir o recurso i. O objetivo do problema é encontrar um subconjunto $A \subseteq F$ que minimize o custo de abrir os recursos em A e atender todos os clientes.*

O problema de localização de recursos apresenta algumas variantes, e dentre elas podemos destacar:

- **Problema de localização de recursos sem capacidades**: Neste problema, um recurso não possui restrições no número máximo de clientes que podem ser associados a ele. Isto possibilita, por exemplo, uma solução onde apenas um recurso é aberto para atender todos os clientes.

- **Problema de localização de recursos com capacidades**: Nesta variante, um recurso tem um limite máximo de clientes que podem ser atendidos por ele.

Além das variações de um recurso possuir restrição de capacidade, existem outras variações que dizem respeito ao custo de associar um cliente ao recurso. Estes custos podem ser métricos ou não-métrico.

2.3.2 Problema de Particionamento

Particionamento *(Clustering)* é a arte de encontrar grupos em um dado conjunto de objetos tal que os objetos pertencentes ao mesmo grupo são similares entre si, enquanto que objetos em diferentes grupos são tão diferentes quanto possível [16].

O problema de dividir um conjunto de dados em um número pequeno de partições com itens relacionados tem um papel crucial em muitas aplicações para extração de conhecimento e análise de dados, tais como busca na web e classificação dos resultados ou interpretação de dados experimentais em biologia molecular. Problemas de particionamento são muito estudados, na literatura, onde encontramos um grande número de variantes bem como diversos algoritmos. Além disso, problemas de particionamento aparecem em outras áreas do conhecimento com nomes diferenciados. Por exemplo, em biologia particionamento ficou conhecido como taxonomia numérica, em ciências sociais como tipologia, na área de reconhecimento de padrões (em Inteligência Artificial) como aprendizado não supervisionado, em lingüística como *clumping*, em geografia como regionalização, em antropologia como serialização, etc.

Em uma definição superficial do problema de particionamento temos um conjunto de n pontos, que desejamos particionar em subconjuntos, respeitando certas restrições estruturais dos elementos entre e em cada parte, de tal forma que uma dada função objetivo seja minimizada (ou maximizada). Esta função objetivo e as restrições estruturais variam de acordo com a aplicação.

Problema 2.3.2 *Uma instância para o problema de particionamento é uma tripla* $(S, \mathcal{C}_S, c)$*, onde*

- S *é um conjunto de* n *objetos,*

- $\mathcal{C}_S$ *é um conjunto de partições de* S*, não necessariamente dado de forma explícita,*

- $c : \mathcal{C}_S \to \mathbb{Q}^+$ *é uma função de custo de uma partição.*

O objetivo é encontrar uma partição $\mathcal{C} \in \mathcal{C}_S$ *que minimize* $c(\mathcal{C})$*.*

Na análise de dados baseada em partições o objetivo é particionar o conjunto de objetos de maneira que objetos em uma mesma parte sejam mais similares possível, enquanto que em relação a objetos externos à parte sejam mais diferentes possível. Existem muitas maneiras de formular matematicamente as funções objetivo de diversos problemas de particionamento, que consiste em buscar a homogeneidade interna à parte e a separação entre partes. Em seguida vamos citar alguns problemas de particionamento mais conhecidos.

- **mínimo k-clustering**: Possui as mesmas características que o problema 2.3.2 acrescido de uma função de distância $d : S \times S \to \mathbb{N}^+$ que satisfaz a desigualdade triangular. O

objetivo é encontrar uma partição de S em k conjuntos de modo que a maior distância entre dois objetos pertencentes a mesma parte seja minimizada. Assim queremos minimizar $\max_{i\in\{1,\ldots,k\}} \max_{x,y\in C_i} d(x,y)$.

- k**-centros**: Dado um grafo $G = (S, E)$ com uma função d de distância nas arestas, temos como objetivo encontrar k elementos $c_1, c_2, \ldots, c_k$ de S, chamados de centros, onde cada $u \in S$ é associado a c_i se $d(u, c_i) = \min_{1\leq j\leq k} d(u, c_j)$, definindo assim k partes. O custo de um particionamento é dado por $\sum_{i=1}^{k} \sum_{u\in C_i} d_{(u,c_i)}$, e o objetivo é encontrar k centros que minimizem esta somatória.

2.3.3 Problema de Cobertura por Conjuntos

O problema da cobertura por conjuntos *(Set Cover)* é um problema muito conhecido em otimização e surge quando tentamos "adquirir", de forma eficiente, itens que estão empacotados em um conjunto fixo de pacotes, podendo ter o mesmo item em pacotes diferentes. O objetivo é "adquirir" todos os itens pegando o menor número de pacotes, ou, na versão com pesos, o conjunto de pacotes que contêm todos os itens e seja o de menor custo, se os pacotes tiverem custos de aquisição.

Problema 2.3.3 *Uma instância para o problema de Cobertura por Conjuntos Mínima (MinCC) é composta de um conjunto* $E_{\text{SC}} = \{e_1, e_2, \ldots, e_n\}$ *de* n *objetos, uma coleção* $\mathcal{S}_{\text{SC}}$ *de subconjuntos de* E_{SC}*, onde* $\mathcal{S}_{\text{SC}} = \{C_1, \ldots, C_t\}$*, e uma função de custo* $c : \mathcal{S}_{\text{SC}} \to \mathbb{Q}^+$*. O objetivo é encontrar uma subcoleção* $\mathcal{S}_{\text{SC}}{}' \subseteq \mathcal{S}_{\text{SC}}$ *tal que* $\cup_{C_i\in\mathcal{S}_{\text{SC}}{}'} C_i = E_{\text{SC}}$ *e* $\sum_{C_i\in\mathcal{S}_{\text{SC}}{}'} c(C_i)$ *é mínimo.*

2.4 Diferenças entre o Problema de Classificação Métrica e Problemas Correlatos.

No problema de classificação métrica, quando desejamos classificar um conjunto de objetos, temos que levar em consideração as informações que temos sobre o conjunto de dados, que são o relacionamento entre objetos e o relacionamento dos objetos com as possíveis classes. Por exemplo, se no problema de restaurar imagens degeneradas por ruídos a intenção fosse associar um ponto a uma cor, sem observar a relação deste ponto com seus vizinhos, poderíamos mapear este problema para o problema de localização de recursos, onde os pontos são os clientes, e as cores são os recursos. No entanto, se desejamos simplesmente criar partições baseadas na localidade e na cor de cada ponto, ou seja, a informação considerada relevante é a relação de um ponto com seus vizinhos, esta abordagem pode ser mapeada como um problema de particionamento.

Ambas as abordagens para o problema de recuperação de imagens levam em consideração apenas parte das informações disponíveis, causando assim uma atribuição de cores inadequada para os pontos. Quando utilizamos o problema de classificação métrica para modelar a restauração de imagens degeneradas por ruídos, obtemos uma modelagem que leva em consideração tanto o custo de associar um ponto a uma cor quanto o custo de atribuir cores diferentes a dois pontos que pertencem à mesma região da imagem.

Capítulo 3

Algoritmos Baseados em Programação Linear

O Problema de Classificação Métrica (ML) foi apresentado pela primeira vez por Kleinberg e Tardos [17], que mostram uma formulação do problema de classificação métrica uniforme como um programa linear inteiro. Mais recentemente, Chekuri *et al.* [7] apresentaram um programa linear que é aplicado ao problema de classificação métrica de uma forma mais natural. Apresentamos a seguir estas duas abordagens.

3.1 Primeiro Programa Linear para o Problema de Classificação Métrica

Kleinberg e Tardos [17] apresentaram a primeira formulação em um programa linear para o problema de classificação métrica uniforme. Posteriormente, esta formulação é adaptada para o caso geral do problema de classificação métrica.

Como vimos no capítulo 2, podemos interpretar o conjunto de objetos P e a função w da relação entre objetos como um grafo $G = (P, E)$ com pesos nas arestas, onde o conjunto dos vértices do grafo é o conjunto P e a relação entre dois objetos $u, v \in P$ é representada por uma aresta não orientada $e = \{u, v\} \in E$, com peso $w_e = w(u, v)$.

Para definir o programa linear inteiro, Kleinberg e Tardos usaram três tipos de variáveis:

- x_{ui} é uma variável binária que indica se o objeto u está associado à classe i, para cada objeto $u \in P$ e cada classe $i \in L$,
- z_e é uma variável binária, para cada $e = \{u, v\} \in E$, que indica se dois objetos relacionados pela aresta e são associados a classes diferentes,
- z_{ei}, para cada $e = \{u, v\} \in E$ e $i \in L$, expressa o valor absoluto de $x_{ui} - x_{vi}$.

A restrição que cada objeto deve ser associado a exatamente uma classe, é dada por

$$\sum_{i \in L} x_{ui} = 1.$$

A distância entre as classes associadas a dois objetos u e v, relacionados pela aresta $e = \{u, v\} \in E$, no problema de classificação métrica uniforme pode ser expressa em função das variáveis x_{ui} como

$$d(f(u), f(v)) = \frac{1}{2} \sum_{i \in L} |x_{ui} - x_{vi}|.$$

Para representar esta distância é introduzida a variável z_{ei}, que representa o valor absoluto de $x_{ui} - x_{vi}$. No programa linear, isso é representado pelas restrições

$$z_{ei} \geq x_{ui} - x_{vi} \qquad \text{e} \qquad z_{ei} \geq x_{vi} - x_{ui},$$

aproveitando que os custos são todos não negativos e a função objetivo é de minimização. Assim, a variável z_e, que representa se dois objetos vão ser separados, é definida pela restrição

$$z_e = \frac{1}{2} \sum_{i \in L} z_{ei}.$$

A função objetivo é composta por dois termos, o custo de associação, que é representado por

$$\sum_{u \in P, i \in L} c_{ui} x_{ui}$$

e o custo de separação, representado por

$$\sum_{e \in E} w_e z_e.$$

Assim, a formulação em programação linear inteira fica:

$$\begin{aligned}
\min \quad & \sum_{e \in E} w_e z_e + \sum_{u \in P, i \in L} c_{ui} x_{ui} \\
s.a. \quad & \sum_{i \in L} x_{ui} = 1 \qquad \forall u \in P, \\
& z_e = \frac{1}{2} \sum_{i \in L} z_{ei} \quad \forall\, e \in E, \\
& z_{ei} \geq x_{ui} - x_{vi} \quad \forall\, e = \{u, v\} \in E, \quad \forall\, i \in L, \\
& z_{ei} \geq x_{vi} - x_{ui} \quad \forall\, e = \{u, v\} \in E, \quad \forall\, i \in L, \\
& x_{ui} \in \{0, 1\} \qquad \forall\, u \in P, \quad \forall\, i \in L.
\end{aligned} \tag{KTG}$$

O programa linear relaxado da formulação (KTG) pode ser obtido removendo-se as restrições de integralidade $x_{ui} \in \{0,1\}$ e acrescentando-se as restrições $x_{ui} \geq 0$, para todo $u \in P$ e $i \in L$. Observe que não é necessário acrescentar a restrição $x_{ui} \leq 1$, visto que o programa linear possui a restrição $\sum_{i \in L} x_{ui} = 1$ que não permite que as variáveis x_{ui} assumam valores maiores que 1. Com essas alterações obtemos a relaxação do programa linear (KTG).

$$
\begin{array}{lll}
\min & \displaystyle\sum_{e=\{u,v\} \in E} w_e z_e + \sum_{u \in P, i \in L} c_{ui} x_{ui} & \\
s.a. & \displaystyle\sum_{i \in L} x_{ui} = 1 & \forall\, u \in P, \\
& z_e = \dfrac{1}{2} \displaystyle\sum_{i \in L} z_{ei} & \forall\, e \in E, \\
& z_{ei} \geq x_{ui} - x_{vi} & \forall\, e = \{u,v\}, \quad \forall\, i \in L, \\
& z_{ei} \geq x_{vi} - x_{ui} & \forall\, e = \{u,v\}, \quad \forall\, i \in L, \\
& x_{ui} \geq 0 & \forall\, u \in P, \quad \forall\, i \in L,
\end{array}
\qquad \text{(KTR)}
$$

3.1.1 Algoritmo para o caso da métrica uniforme

Seja x uma solução do programa linear inteiro (KTG). Se x_{ui} for igual a 1, significa que, levando em consideração os custos de associação e os custos de separação para o objeto u e seus vizinhos, uma melhor classificação para o objeto u é a classe i.

No programa linear relaxado (KTR), a variável x_{ui} pode assumir valor fracionário entre 0 e 1. No entanto, relembrando o exemplo de recuperação de imagens degeneradas por ruídos, não podemos dizer que um ponto será 30% verde, 40% vermelho e 30% azul. Então, surge a necessidade de transformar os valores fracionários das variáveis x_{ui} em valores inteiros, para tanto, o conteúdo de uma variáveis x_{ui} é interpretado como a probabilidade do objeto u ser atribuído à classe i.

Uma primeira idéia que surge é considerar cada variável x_{ui} separadamente, sortear um valor aleatório pertencente ao intervalo $[0,1]$ e decidir, baseado neste valor aleatório, se aquele objeto será associado à classe. Entretanto, nesta abordagem pode ocorrer de objetos relacionados, com probabilidades iguais ou semelhantes, serem separados.

Kleinberg e Tardos [17] apresentam um algoritmo, que denominamos de *Alg_UML*, iterativo dois aproximado para o arredondamento do resultado do programa linear (KTR). Em cada iteração é escolhida uma classe para ser analisada. Veja o algoritmo na figura 3.1.

Para melhor ilustrar o funcionamento do algoritmo *Alg_UML* utilizaremos uma instância para o problema de classificação métrica uniforme, apresentada na figura 3.2, que contendo duas classes e três objetos. Assumimos a solução apresentada na tabela 3.1 como uma solução do programa linear (KTR) para esta instância, onde o conteúdo da célula na linha u coluna i representa o valor da variável x_{ui}.

ALGORITMO *Alg_UML* (P, L, d, c, w)
onde (P, L, d, c, w) é uma instância para o problema UML.
1. *% Inicialmente nenhum objeto possui uma classe associada*
2. Seja (x, z) uma solução para o programa linear (KTR)
3. Executa *Round_UML*(P, L, x)

PROCEDIMENTO *Round_UML*(P, L, x)
1. $U \leftarrow P$
2. Enquanto $U \neq \emptyset$
3. Selecione aleatoriamente uma classe $i \in L$
4. $\alpha \leftarrow random[0, 1]$
5. Para todo $u \in U$
6. Se $\alpha \leq x_{ui}$ então
7. $f(u) \leftarrow i$
8. $U \leftarrow U \setminus \{u\}$
9. Devolva f

Figura 3.1: Algoritmo *Alg_UML* para o arredondamento da solução do programa linear (KTR) para o problema UML.

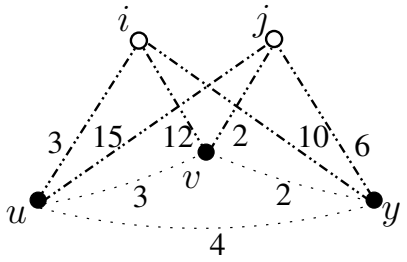

Figura 3.2: Exemplo de uma instância.

	i	j
u	0,6	0,4
v	0,2	0,8
y	0,5	0,5

Tabela 3.1: Os valores de x para uma solução do programa linear (KTR) para a instância da figura 3.2.

Inicialmente o algoritmo seleciona aleatoriamente uma classe e sorteia um valor α no intervalo $[0, 1]$. Vamos assumir que i é a classe selecionada e o valor de α é 0,4. Em seguida, o

algoritmo verifica quais são os objetos não classificados (representados pelo conjunto U) com valor x_{ui}, $u \in U$, maior que α. Pela tabela 3.1, verificamos que esta condição é satisfeita para os objetos u e y. Assim os objetos u e y são associados à classe i e removidos do conjunto U. O algoritmo repete este procedimento até que todos os objetos tenham sido classificados.

Note que a classe i pode ser escolhida novamente em uma próxima iteração e ter o objeto v associado à ela, No entanto, é mais provável que o objeto v seja associado à classe j uma vez que x_{vi} é menor que x_{vj}.

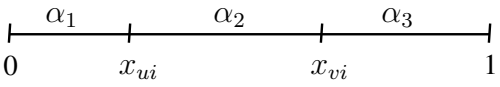

Figura 3.3: Possíveis situações na escolha de α.

Para compreender melhor a função da variável α no algoritmo *Alg_UML*, observe a figura 3.3, onde temos três casos que podem ocorrer em relação ao valor de α e a separação de objetos u e v relacionados:

a) O valor de α ocorre no intervalo α_1. Neste caso os objetos u e v são associados à mesma classe.

b) O valor de α ocorre no intervalo α_2. Neste caso apenas o objeto u tem sua classe associada nesta iteração e dizemos que u e v são separados.

c) O valor de α ocorre no intervalo α_3. Neste caso os objetos u e v não são associados à classe i nesta iteração.

Os lemas a seguir apresentam propriedades do algoritmo probabilístico *Alg_UML*.

Lema 3.1.1 *Em uma iteração do* Enquanto *do procedimento Round_UML, a probabilidade de um objeto $u \in U$ ser associado a uma classe i é precisamente x_{ui}/m e a probabilidade de u ser associado a qualquer classe em uma iteração é $1/m$, onde m é o número de classes. Assim, numa execução do Round_UML, a probabilidade do objeto u ser associado à classe i é exatamente x_{ui}.*

Prova. Pela restrição $\sum_{i \in L} x_{ui} = 1$ temos que a probabilidade de um objeto u ser associado a qualquer classe é 1. A probabilidade de uma classe ser escolhida é $1/m$. Assim, a probabilidade de um objeto u ser associado a uma classe qualquer em uma iteração é igual a $1/m$.

Em cada iteração, enquanto o objeto u não tem uma classe associada, a probabilidade dele ser associado a uma classe i é $\frac{1}{m}x_{ui}$. Assim a probabilidade de um objeto u ser associado a uma classe i pelo algoritmo é precisamente x_{ui}.

□

Sabemos que para ocorrer a separação de dois objetos relacionados, eles devem ser associados a alguma classe em iterações diferentes. O lema a seguir dá a probabilidade de dois objetos serem associados a classes em iterações diferentes.

Lema 3.1.2 *Em uma iteração, a probabilidade de dois objetos u e v em U, relacionados pela aresta e, serem separados é $2z_e/m = \sum_{i \in L} |x_{ui} - x_{vi}|/m$. A probabilidade dos dois objetos u e v, relacionados pela aresta $e = \{u, v\}$, serem separados pelo algoritmo é no máximo $2z_e$.*

Prova.

Considerando uma classe i escolhida na linha 3 do procedimento *Round_UML*, a probabilidade de somente um dos objetos u e v em U ser associado à classe i é $|x_{ui} - x_{vi}|$. Então, a probabilidade de somente um dos objetos u e v, relacionados pela aresta $e = \{u, v\}$, ser associado a alguma classe em uma iteração é

$$\sum_{i \in L} \frac{1}{m} |x_{ui} - x_{vi}| = \frac{\sum_{i \in L} z_{ei}}{m} = \frac{2z_e}{m}.$$

Se os objetos u e v, relacionados pela aresta $e = \{u, v\}$, forem associados a uma classe na mesma iteração, então o custo de separação w_e, onde $e = \{u, v\}$, não será pago. Isto é, em cada iteração o algoritmo escolhe aleatoriamente uma classe $i \in L$ e um valor α entre 0 e 1. Se x_{ui} é maior ou igual a α então u é associado à classe i. Seja $\mathcal{E}$ o evento em que u e v são separados em uma iteração e $\mathcal{E}'$ o evento em que pelo menos um de u e v é associado em uma iteração. Podemos obter uma delimitação superior da probabilidade de dois objetos relacionados serem separados pelo processo do seguinte modo:

$$\frac{\Pr[\mathcal{E}]}{\Pr[\mathcal{E}']}. \tag{3.1}$$

A probabilidade de ocorrer o evento $\mathcal{E}$ é $2z_e/m$ e a probabilidade de ocorrer o evento $\mathcal{E}'$ é pelo menos a probabilidade do objeto u ser associado que é, pelo lema 3.1.1, $1/m$. Então, a probabilidade dos objetos u e v, relacionados pela aresta $e = \{u, v\}$, serem associados a classes diferentes é no máximo $2z_e$.

□

Com o auxílio dos lemas anteriores podemos enunciar o próximo teorema que nos dá o fator de aproximação do algoritmo.

Teorema 3.1.3 *Seja x uma solução do programa linear relaxado (KTR) com c_{LP} representando o custo de associação e w_{LP} representando o custo de separação da correspondente solução. O algoritmo obtém uma classificação x onde o custo de associação esperado é c_{LP}, e o custo de separação esperado é no máximo $2w_{LP}$.*

Prova.

Pelo lema 3.1.1, a probabilidade $\Pr[f(u) = i]$ é x_{ui}. Então, a contribuição esperada de u para a função objetivo é $\sum_{i \in L} c_{ui} x_{ui}$, que é exatamente igual à contribuição de u no programa linear. Assim, o custo total de associação esperado é de c_{LP}.

A contribuição esperada da aresta $e = \{u, v\}$ para o custo de separação é exatamente w_e vezes a probabilidade dos objetos u e v serem separados, que pelo lema 3.1.2 é no máximo $2z_e$. A contribuição esperada da aresta e é no máximo $2w_e z_e$, que é duas vezes a contribuição do programa linear. Portanto, o custo total esperado de associação sobre todos os objetos é no máximo $2w_{LP}$.

□

Corolário 3.1.4 *Usando uma solução x do programa linear (KTR) obtemos uma classificação com valor esperado que é no máximo duas vezes o valor de uma solução ótima.*

Lema 3.1.5 *O tempo de execução esperado do algoritmo Alg_UML é no máximo m.*

Prova.

O tempo de execução esperado é dado por $\sum_t t\Pr(t)$, onde t é o tempo de execução e $\Pr(t)$ é a probabilidade do tempo de execução t ocorrer.

Pelo lema 3.1.1, a probabilidade de um objeto ser associado a qualquer classe em uma iteração é $\frac{1}{m}$. Note que a probabilidade de um objeto ser associado a uma classe qualquer em uma iteração não é alterada pelo fato de outros objetos terem sido associados, já que os valores de x_{ui} não são alterados durante a execução do algoritmo. Assim, a probabilidade de um objeto ser associado na i-ésima iteração é dado por $1/m(1 - 1/m)^i$, e o tempo de execução esperado do algoritmo é dado por

$$\begin{aligned}
\sum_{i=1}^{\infty} i \left[\left(1 - \frac{1}{m}\right)^{i-1} \frac{1}{m} \right] &= \frac{1}{m} \sum_{i=1}^{\infty} i \left(1 - \frac{1}{m}\right)^{i-1} \\
&= \frac{\frac{1}{m}}{\left(1 - \frac{1}{m}\right)} \sum_{i=1}^{\infty} i \left(1 - \frac{1}{m}\right)^{i} \\
&= \frac{\frac{1}{m}}{\left(1 - \frac{1}{m}\right)} \left[\frac{\left(1 - \frac{1}{m}\right)}{\left(1 - \left(1 - \frac{1}{m}\right)\right)^2} \right] \\
&= \frac{\frac{1}{m}}{1 - 2\left(1 - \frac{1}{m}\right) + \left(1 - \frac{1}{m}\right)^2} \\
&= \frac{\frac{1}{m}}{\frac{1}{m^2}} \\
&= \frac{m^2}{m}
\end{aligned}$$

$$= \quad m.$$

Assim, o tempo de execução esperado do algoritmo *Alg_UML* é no máximo m.

□

3.1.2 Árvores métricas

Antes de apresentarmos a abordagem considerada por Kleinberg e Tardos [17] para o caso da métrica geral apresentaremos os resultados obtidos por Bartal [2, 3] sobre a aproximação de espaços métricos através de árvores métricas. Assumimos que um espaço métrico $M^V = (M, V)$ é a aplicação da métrica M sobre o conjunto de pontos V.

O problema de aproximação de espaços métricos depende diretamente da escolha da classe de métricas que tomamos como sendo "simples". Tipicamente é mais fácil obter bons resultados em problemas de otimização quando os dados respeitam alguma métrica mais "simples". Um exemplo de métrica simples é a métrica uniforme, além da métrica uniforme vale destacar as métricas geradas por árvores (também conhecidas como métricas aditivas).

Seja N^V e M^V dois espaços métricos sobre um conjunto finito de pontos V, dizemos que o espaço métrico N^V *domina* o espaço métrico M^V se para todo par de pontos $u, v \in V$, $d_M(u, v) \leq d_N(u, v)$. Seja $\mathcal{S}$ um conjunto de espaços métricos "simples" sobre um conjunto de pontos V. O espaço métrico M^V é dito ser α-aproximado-probabilisticamente por $\mathcal{S}$ se existir uma distribuição de probabilidade em $\mathcal{S}$, onde para cada par de vértices $u, v \in V$ a esperança da distância em um espaço métrico N^V, escolhida probabilisticamente em $\mathcal{S}$, é no máximo α vezes maior que a distância em M^V, ou seja $E(d_N(u, v)) \leq \alpha d_M(u, v)$.

Se temos uma β-aproximação para um problema de otimização, onde o espaço métrico, sobre o conjunto de dados V, pertence à classe $\mathcal{S}$, e se um espaço métrico $N^V \notin \mathcal{S}$ pode ser α-aproximado-probabilisticamente por $\mathcal{S}$ então, a razão de performance dos algoritmos para um espaço métrico $N^V \notin \mathcal{S}$ é $\alpha\beta$.

Um espaço métrico M^V sobre um conjunto de pontos V pode ser visto como um grafo $G_M = (V, E, w)$, onde um peso $w(e)$, para uma aresta $e \in E$, representa a distância entre os pontos ligados pela aresta e. Um exemplo de espaço métrico simples para M^V é o espaço métrico gerado pela árvore geradora de peso mínimo de G_M, onde a distância entre dois elementos $u, v \in V$ é dada pela soma das arestas do caminho mínimo de u até v na árvore geradora de peso mínimo. Este espaço métrico possui uma distorção na distância entre os pontos de no máximo $\Omega(n)$ [2], onde n é o número de pontos em V.

Bartal [2] explora o fato de que os espaços métricos que formam a classe $\mathcal{S}$ e são usados para aproximar uma métrica M, definida sobre um conjunto de pontos V, não precisam ser induzidos por um subgrafo do grafo gerado pela métrica M. Isto permite considerar uma classe onde os espaços métricos são gerados por árvores métricas e possuem outras características desejáveis.

As árvores métricas que compõe uma classe de espaços métricos são chamadas de árvores r-hierarquicamente bem separadas (r-*hierarchically-well-separated trees* r-HST). Temos como propriedade das árvores métricas r-HST que o espaço métrico gerado pode ser decomposto recursivamente em subespaços métricos, e para um espaço métrico com diâmetro Δ os seus subespaços métricos terão diâmetros menores em um fator de r, ou seja, Δ/r, Δ/r^2, Δ/r^3, . . ., para um $r > 1$. A figura 3.4 ilustra um exemplo de uma árvore r-HST onde podemos observar as propriedades destas árvores.

Mais formalmente podemos definir uma árvore r-HST da seguinte forma.

Definição 3.1.6 *Uma árvore métrica r-hierarquicamente bem separada (r-HST) é definida como uma árvore enraizada onde as arestas possuem pesos com as seguinte propriedades:*

- *As arestas que ligam um pai aos seus filhos tem o mesmo peso.*
- *Os pesos das arestas ao longo de qualquer caminho que vai da raiz até uma folha decrescem por um fator maior que 1, que é uma potência de r, ou seja, se e_{vp} e e_{vf} são arestas que ligam um vértice v ao seu pai e v a um de seus filhos, respectivamente, então $w(e_{vf}) = w(e_{vp})/r^t$, para algum $t >= 1$.*
- *A distância entre dois nós da árvore é dada pela soma do tamanho das arestas do único caminho entre estes nós na árvore.*

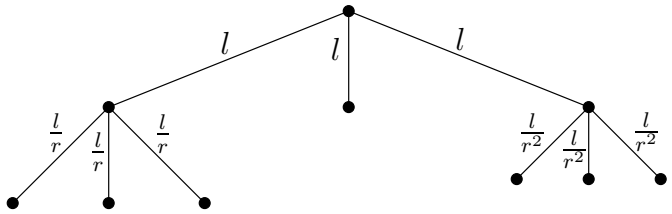

Figura 3.4: Exemplo de uma árvore r-HST.

Note que a distância entre pontos irmãos em uma árvore r-HST é uniforme. Isto permite a utilização da técnica de divisão e conquista para obter uma solução para um problema de otimização definido sobre qualquer métrica M.

O principal resultado de Bartal [2, 3] é apresentado no teorema 3.1.7

Teorema 3.1.7 *Todo grafo conexo G com pesos nas arestas pode ser α-aproximado-probabilisticamente por um conjunto de árvores r-HST, que possuem um diâmetro proporcional ao diâmetro do grafo G, em tempo polinomial, onde $\alpha = O(r \log n \log \log n)$.*

A construção de uma árvore r-HST que aproxima um espaço métrico M^V, definida sobre um conjunto de vértices V, é baseada em uma partição métrica hierárquica *(Hierarchical Partition Metric (HPM))* do grafo G_M.

Uma partição métrica hierárquica é uma partição recursiva do grafo G_M, onde em cada iteração o grafo é particionado e as arestas que definem a partição são rotuladas com um valor maior que o diâmetro do subgrafo considerado na iteração. A partição e a atribuição de rótulos é repetida para cada parte obtida. Como resultado da HPM temos uma estrutura muito próxima de uma árvore HST.

A construção de uma HPM para um grafo ponderado $G = (V, E, w)$ é feita recursivamente como mostra o algoritmo *Alg_HPM* da figura 3.5. O algoritmo *Alg_HPM* utiliza como entrada um grafo com pesos nas arestas e devolve uma atribuição de rótulos ℓ para as arestas.

ALGORITMO *Alg_HPM* (G)

onde $G = (V, E, w)$ é um grafo ponderado.

1. $U = V$
2. $\phi = \phi(G) = |E|$
3. $\Delta = diam(G)$
4. $T = \phi = |E(G)|$.
5. Seja $\widetilde{G}$ o grafo obtido a partir de G contraindo as arestas em $\{e \in E | w(e) < \frac{\Delta}{2T}\}$
6. Seja $\widetilde{\Delta} = \Delta/2$.
7. Escolha aleatoriamente um vértice $v \in U$.
8. Escolha aleatoriamente um inteiro g no intervalo $[0, T]$.
9. Defina o raio $z_g = \frac{g}{T}\widetilde{\Delta}$.
10. Defina um corte $C_g = \{\{u, w\} \in E(\widetilde{G}) | d(v, u) \leq z_g < d(v, w)\}$.
11. Para cada $e \in C_g$
12. $\quad \ell(e) = \Delta$
13. Seja $U_g^1 = \{u \in V(\widetilde{G}) | d(u, v) < z_g\}$ e $U_g^2 = \{u \in V(\widetilde{G}) | d(u, v) \geq z_g\}$.
14. Seja H_g^1 e H_g^2 os subgrafos induzidos de G pelos conjuntos de vértices U_g^1 e U_g^2.
15. Se $U_g^1 \neq \emptyset$ então
16. $\quad$ Execute *Alg_HPM*(H_g^1)
17. Se $U_g^2 \neq \emptyset$ então
18. $\quad$ Execute *Alg_HPM*(H_g^2)

Figura 3.5: Algoritmo para a partição métrica hierárquica.

Como podemos ver na figura 3.6, o resultado da partição métrica hierárquica é uma atribuição de rótulos ℓ para as arestas do grafo G. Os rótulos atribuídos para as arestas são representados por tipos de linhas diferentes. Na figura 3.6(a) temos o grafo G e na figura 3.6(b) temos o resul-

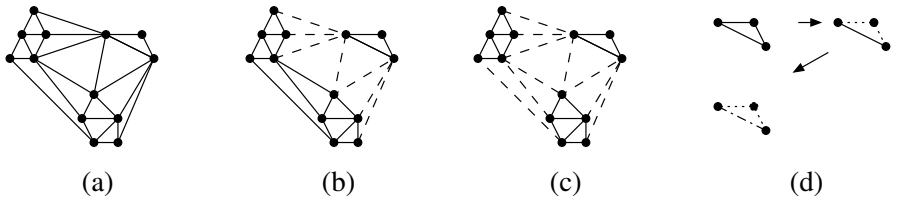

Figura 3.6: Exemplo de uma partição métrica hierárquica.

tado da primeira iteração do algoritmo. Claramente o conjunto de vértices U_g^2 é composto pelos vértices do componente de maior tamanho da figura 3.6(b). Aplicando HPM para o subgrafo induzido por U_g^2 temos a figura 3.6(c). Observe que as arestas rotuladas da figura 3.6(c) possuem o mesmo rótulo. Isto ocorre porque o diâmetro do subgrafo formado pelo conjunto U_g^2 é igual ao diâmetro do grafo G. Aplicando recursivamente o algoritmo *Alg_HPM* para o subgrafo formado pelo conjunto U_g^1 temos a atribuição ilustrada na figura 3.6(d).

Dada uma partição métrica hierárquica ℓ de um grafo G, o algoritmo *Alg_HST*, ilustrado na figura 3.7, constrói uma árvore r-HST, onde cada vértice da árvore r-HST representa um subgrafo da partição métrica de G.

Analisando as distorções geradas na criação da HPM e na construção da árvore r-HST, temos que para qualquer métrica M, definida sobre um conjunto V de n pontos, podemos construir uma árvore métrica r-HST, que é uma $O(\log n \log\log n)$-aproximação da métrica M, em tempo polinomial.

Recentemente Fakcheroenphol *et al.* [9] melhoraram o resultado de Bartal apresentando um algoritmo que aproxima um espaço métrico qualquer por árvores métricas HST com fator de aproximação $O(\log n)$.

3.1.3 Algoritmo para o caso da métrica geral (Problema ML)

Observe que, utilizando o algoritmo *Alg_UML*, nem sempre se obtém uma solução, com garantia no fator de aproximação, para instâncias do problema de classificação métrica com distâncias variadas. Isto porque o programa linear (KTR) não considera em sua formulação o valor da função de distância.

Kleinberg e Tardos [17], utilizando o resultado obtido por Bartal [2, 3], anunciado no teorema 3.1.7, desenvolveram uma variação do algoritmo *Alg_UML* e do programa linear (KTR), que obtém soluções $O(\log m \log\log m)$-aproximadas para instâncias do problema de classificação métrica.

A idéia básica do algoritmo consiste em aproximar o espaço métrico definido pela função de distância d do conjunto de classes L através de uma árvore enraizada r-HST, onde r é um inteiro maior que 2, e utilizar esta aproximação do espaço métrico como um dos parâmetros de entrada para um programa linear, semelhante ao apresentado para o problema de classificação

ALGORITMO *Alg_HST* (G, ℓ, r, L, k)
onde $G = (V, E, w)$ é um grafo ponderado,
ℓ é uma partição métrica hierárquica de G,
r é um fator de decréscimo da distância,
L é o diâmetro do subgrafo considerado, inicialmente $diam(G)$ e
k é a raiz a quem será ligado a subárvore, inicialmente NILL.

1. Se $k =$ NILL faça.
2. Adicione um vértice v_G como sendo a raiz.
3. Aponte v para o vértice v_G.
4. Seja $\{H_1, \ldots, H_m\}$ os m subgrafos de G gerados no primeiro nível da HPM ℓ.
5. Para $i = 1$ até m faça
6. Se $H_i \neq \emptyset$ então
7. Execute *Alg_HST*(H_i, ℓ, r, L, v)
8. Senão
9. Crie um vértice v_G.
10. Conecte o vértice v_G como filho de k com uma aresta de comprimento $L/2$.
11. Se $diam(G) < L/r$ então
12. Aponte v para o vértice v_G.
13. Decremente L por um fator de r.
14. Senão
15. Aponte v para o vértice k.
16. Seja $\{H_1, \ldots, H_m\}$ os m subgrafos de G gerados no primeiro nível da HPM ℓ.
17. Para $i = 1$ até m faça
18. Se $H_i \neq \emptyset$ então
19. Execute *Alg_HST*(H_i, ℓ, r, L, v)

Figura 3.7: Algoritmo para a construção de uma árvore r-HST.

uniforme.

Kleinberg e Tardos acrescentaram três novas restrições, para que uma árvore r-HST, sobre o espaço métrico das classes, possa ser utilizada pelo novo programa linear e pelo novo algoritmo. São elas:

a) O parâmetro r, utilizado na construção da árvore r-HST, tem que ser maior que 2.

b) Todos os nós que representam as classes devem ser folhas.

c) Os nós internos da árvore, nós que não são folhas, são apenas auxiliares.

Note que ao construir uma árvore r-HST para o espaço métrico formado pelo conjunto de classes L de uma instância do problema de classificação métrica uniforme, obtemos uma árvore em formato de estrela, onde todas as folhas são classes. O nó central é um nó auxiliar e o tamanho de todas as arestas é a metade da distância entre quaisquer duas classes, que é um valor constante.

Seja T uma árvore r-HST, onde as classes são folhas de T, e T' uma subárvore qualquer de T. Denotamos por $L(T')$ o conjunto das classes que são folhas de T'. Vamos usar a notação $T' < T$ para árvores T' que são subárvores de T. Assim, $\cup_{T'<T} L(T') = L$.

Após construirmos uma árvore T, r-HST, para as classes em L, podemos formular o problema como um programa linear inteiro, semelhante ao programa linear (KTG), utilizando as mesmas variáveis x_{ui} e z_e utilizadas anteriormente. A principal diferença está na inclusão de variáveis $x_{uT'}$, para $T' < T$, que é uma variável auxiliar no cálculo do custo de separação e estabelecer que

$$x_{uT'} = \sum_{i \in L(T')} x_{ui}.$$

Esta variável indica se a classe a qual o objeto u será associado pertence à subárvore T', onde T' é uma subárvore enraizada em algum nó, inicialmente a raiz.

Dada uma subárvore T' de T cuja raiz é y, denotamos por $\ell_{T'}$ o comprimento da aresta $\{y, s\}$, onde s é o pai de y em T. Como conseqüência da definição da variável x_{ui}, que indica se o objeto u será associado à classe i, a variável $x_{uT'}$ indica se o objeto u será associado a uma classe que pertence a $L(T')$. Assim, a distância entre as classes associadas aos objetos u e v, relacionados pela aresta $e = \{u, v\}$, é definida como

$$d(f(u), f(v)) = \sum_{T'<T} \ell_{T'} |x_{uT'} - x_{vT'}|.$$

Para auxiliar na construção do programa linear temos a variável $z_{eT'}$ que expressa o valor absoluto $|x_{uT'} - x_{vT'}|$. A seguir apresentaremos o programa linear inteiro do problema de classificação métrica usando uma árvore T, r-HST:

$$\begin{array}{lll}
\min & \displaystyle\sum_{e \in E} w_e z_e + \sum_{u \in P, i \in L} c_{ui} x_{ui} & \\
s.t. & \displaystyle\sum_{i \in L} x_{ui} = 1 & \forall\, u \in P, \\
& z_e = \displaystyle\sum_{T'<T} \ell_{T'} z_{eT'} & \forall\, e \in E, \\
& z_{eT'} \geq x_{uT'} - x_{vT'} & \forall\, e = \{u, v\} \in E,\ \ T' < T, \\
& z_{eT'} \geq x_{vT'} - x_{uT'} & \forall\, e = \{u, v\} \in E,\ \ T' < T, \\
& x_{uT'} = \displaystyle\sum_{i \in L(T')} x_{ui} & T' < T, \\
& x_{ui} \in \{0, 1\} & \forall\, u \in P, \quad \forall\, i \in L.
\end{array} \qquad \text{(KTTG)}$$

Assim como no programa linear (KTG), obtemos o programa linear relaxado fazendo com que as variáveis x_{ui} recebam valores no intervalo $[0,1]$, ao invés dos valores binários $\{0,1\}$, e acrescentamos a restrição $x_{ui} \geq 0$ no programa linear (KTTG). Baseado nestas alterações obtemos o seguinte programa linear:

$$
\begin{array}{lll}
\min & \displaystyle\sum_{e=\{u,v\}\in E} w_e z_e + \sum_{u\in P, i\in L} c_{ui} x_{ui} & \\
s.t. & \displaystyle\sum_{i\in L} x_{ui} = 1 & \forall\, u \in P, \\
& z_e = \displaystyle\sum_{T'<T} \ell_{T'} z_{eT'} & \forall\, e \in E, \\
& z_{eT'} \geq x_{uT'} - x_{vT'} & \forall\, e=\{u,v\} \in E,\ T' < T, \\
& z_{eT'} \geq x_{vT'} - x_{uT'} & \forall\, e=\{u,v\} \in E,\ T' < T, \\
& x_{uT'} = \displaystyle\sum_{i\in L(T')} x_{ui} & T' < T, \\
& x_{ui} \geq 0 & \forall\, u \in P, \quad \forall\, i \in L.
\end{array}
\qquad \text{(KTT)}
$$

No programa linear relaxado (KTT), a variável x_{ui} pode assumir um valor fracionário entre 0 e 1, que o algoritmo interpreta como a probabilidade do objeto u ser atribuído à classe i. Pela definição da variável x_{ui}, a variável x_{uT} pode ser interpretada como a probabilidade do objeto u ser associado a alguma classe pertencente ao conjunto formado pelas folhas da subárvore T.

O algoritmo *Alg_ML*, apresentado na figura 3.8, utiliza como ponto de partida uma solução do programa linear relaxado (KTT) e considera inicialmente as t subárvores, que tem como raiz um filho da raiz de T, dadas por $T^1, \ldots, T^t$ (assumindo que a raiz de T possui t filhos). Em seguida são passados como parâmetros para a execução do procedimento *Round_UML* o conjunto de objetos P, um conjunto de classes, onde cada uma das t subárvores é considerada uma classe, e os valores x_{uT} respectivos às subárvores consideradas na iteração.

Como resultado da primeira iteração temos a associação de um conjunto de objetos $P' \subseteq P$ a cada classe representada pela subárvore T^j, onde $j \in \{1, \ldots, t\}$. Posteriormente assumimos $\overline{x}_{ui} = x_{ui}/x_{uT^j}$, para todas as classes $i \in L(T^j)$ e para todo $u \in P$, como uma normalização dos valores de x_{ui}, para que a soma dos valores de $\overline{x}_{ui}$ em uma subárvore T^j seja igual a um e possam ser utilizados na chamada recursiva para a subárvore T^j, onde $j \in \{1, \ldots, t\}$.

Na figura 3.9 temos uma instância para o problema de classificação métrica. A figura 3.9(a) ilustra uma árvore 3-HST que será utilizada pelo algoritmo *Alg_ML* e na figura 3.9(b) temos o grafo formado pelos objetos e suas relações, já na figura 3.9(c) temos a decomposição da árvore T em subárvores.

Os valores das arestas que ligam as subárvores de T aos seus pais são: $\ell(T_1) = 3$, $\ell(T_2) = 3$, $\ell(T_3) = 1$ e $\ell(T_4) = 1$. Os custos de atribuição estão dispostos na tabela ilustrada na figura 3.10, onde o valor da célula na linha a coluna b representa o custo de associação c_{ab}.

ALGORITMO *Alg_ML* (P, L, d, c, w, T)

onde (P, L, d, c, w) é uma instância para o problema ML e
T é uma árvore r-HST que aproxima a métrica de L.

1. Seja x uma solução para o programa linear (KTT)
2. Execute *Round_ML*(P, T, x)
3. Devolva f

PROCEDIMENTO *Round_ML* (P, T, x)

1. $y_{uj} = x_{uT^j}\ \ \forall\ 1 \leq j \leq t$ e $\forall\ u \in P$, onde T^j é filho de T
2. Execute *Round_UML*(P, $\{T^1, \ldots, T^t\}$, y)
3. Para $j = 1$ até t faça
4. $\quad P_{T^j} \leftarrow$ objetos associados à subárvore T^j
5. $\quad$ Se a altura de T^j é igual a zero então
6. $\quad\quad$ Para todo $u \in P_{T^j}$ faça
7. $\quad\quad\quad f(u) = L(T^j)$
8. $\quad$ Senão
9. $\quad\quad$ Para todo $i \in L(T^j)$ e $u \in P$
10. $\quad\quad\quad \overline{x}_{ui} \leftarrow x_{ui}/x_{uT^j}$
11. $\quad\quad$ Para todo $T^l < T^j$
12. $\quad\quad\quad \overline{x}_{uT^l} \leftarrow \sum_{i \in L(T^l)} \overline{x}_{ui}$
13. $\quad\quad$ Execute *Round_ML*(P_{T^j}, T^j,$\overline{x}$)

Figura 3.8: Algoritmo para o arredondamento da solução do programa linear para o caso métrico.

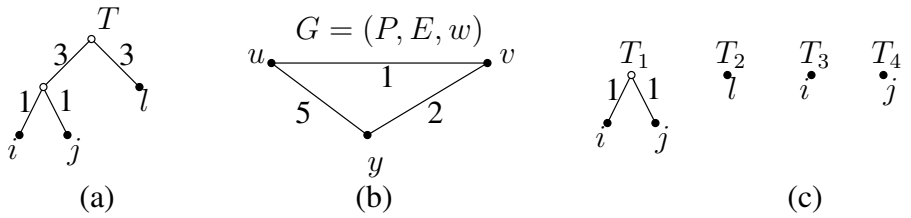

Figura 3.9: Exemplo de uma instância para o algoritmo *Alg_ML*.

Para melhor ilustrar o funcionamento do algoritmo *Alg_ML* considere a instância ilustra na figura 3.9 com os valores de x, de uma solução do programa linear (KTT), apresentados na tabela da figura 3.11, onde o conteúdo de uma célula ui representa o valor de x_{ui}.

O algoritmo *Alg_ML* executa o procedimento *Round_ML* passando como parâmetros o conjunto P, a árvore T e a atribuição de x, representado pela tabela da figura 3.11. O procedimento

	i	j	l
u	2	3	6
v	6	5	2
y	4	4	3

Figura 3.10: Custo de associação da instância ilustrada na figura 3.9.

	i	j	l	T_1	T_2	T_3	T_4
u	0,6	0,2	0,2	0,8	0,2	0,6	0,2
v	0,2	0	0,8	0,2	0,8	0,2	0
y	0,4	0,4	0,2	0,8	0,2	0,4	0,4

Figura 3.11: Solução do programa linear (KTT) para a instância da figura 3.9.

Round_ML executa o procedimento *Round_UML* passando como parâmetros o conjunto de objetos P e o conjunto $\{T^1, \ldots, T^m\}$ de classes, onde $T^1, \ldots, T^m$ são filhos de T. Para o nosso exemplo o conjunto de classes $L = \{T_1, T_2\}$, e os valores de x passados para o procedimento são as colunas T_1 e T_2 da tabela 3.11.

Supondo que a solução do procedimento *Round_UML* seja a atribuição dos objetos u e y à subárvore T_1 e o objeto v à subárvore T_2. O procedimento *Round_ML* é executado recursivamente para a subárvore T_1 com o conjunto de objetos $\{u, y\}$ e para a subárvore T_2 com o conjunto de objetos $\{v\}$. Os valores de x são atualizados para cada subárvore. Para a subárvore T_1 os valores de x considerados são os representados pela tabela 3.2; O procedimento *Round_ML* não é executado para a subárvore T_2 devido a esta não possuir filhos.

	i	j	T_3	T_4
u	0,75	0,25	0,75	0,25
y	0,5	0,5	0,5	0,5

Tabela 3.2: Os valores de x atualizados para a subárvore T_1.

Ao executarmos o procedimento *Round_ML* para a subárvore T_1 com seus respectivos objetos e valores de x, este executa o procedimento *Round_UML* que devolve uma classificação dos objetos u e y em uma das subárvores T_3 ou T_4, concluindo assim a execução do algoritmo *Alg_ML*.

Para analisar este algoritmo são usadas as propriedades válidas para a métrica uniforme, que foram provadas nos lemas 3.1.1 e 3.1.2, e estão adaptadas, no lema 3.1.8, para métricas aproximadas através de árvores r-HST.

Lema 3.1.8 *A probabilidade do objeto u ser associado a uma subárvore T^j é x_{uT^j}. Para cada aresta $e = \{u, v\}$ a probabilidade dos objetos u e v serem associados a classes diferentes é no máximo*

$$\sum_{j=1}^{t} |x_{uT^j} - x_{vT^j}|.$$

Quando consideramos um conjunto de objetos, onde todos são associados à mesma subárvore, temos o problema que o seu custo de separação fracionário é alterado devido à normalização aplicada sobre as variáveis x_{ui}. Como veremos, o aumento no custo de separação resultante desta alteração é limitado no lema 3.1.10. Antes de mostrarmos este limite precisamos de alguns fatos importantes. A prova é baseada na soma de todos os níveis da árvore e é usada a propriedade das árvores r-HST de que o comprimento das arestas na árvore decresce exponencialmente conforme andamos em direção às folhas.

Lema 3.1.9 *Seja x uma solução fracionária devolvida pelo programa linear (KTT) quando aplicado a uma instância do problema ML, com o espaço métrico do conjunto de classes L aproximado através de uma árvore r-HST com $r > 2$, cujas folhas são classes. Então, para um objeto u e uma subárvore T,*

$$\sum_{T' < T} \ell_{T'} x_{uT'} \leq \frac{1}{r-1} \ell_T x_{uT}.$$

Prova.

Se t é a raiz de T e t' é a raiz de uma subárvore T' de T, dizemos que T' está no nível j se o caminho que parte de t e chega em t' tem exatamente j arestas e denotamos por $D(T', T)$ o nível da subárvore T' em T. Note que temos

$$\sum_{T' : D(T',T)=j} x_{uT'} \leq x_{uT}$$

para todo $j \geq 1$. Assim

$$\begin{aligned}
\sum_{T' < T} \ell_{T'} x_{uT'} &= \sum_{j \geq 1} \sum_{T' : D(T',T)=j} \ell_{T'} x_{uT'} \\
&\leq \sum_{j \geq 1} r^{-j} \ell_T \sum_{T' : D(T',T)=j} x_{uT'} \\
&\leq \sum_{j \geq 1} r^{-j} \ell_T x_{uT} \\
&\leq \frac{1}{r-1} \ell_T x_{uT}.
\end{aligned}$$

□

Lema 3.1.10 *Seja $e = \{u, v\}$ uma aresta e suponha que $x_{uT^i} \leq x_{vT^i}$. Se ambos, u e v, são associados à subárvore T^i então o novo custo de separação fracionário é no máximo*

$$\frac{w_e}{x_{uT^i}} \left[\sum_{T' < T^i} \ell_{T'} |x_{uT'} - x_{vT'}| + \frac{1}{r-1} \ell_{T^i} |x_{uT^i} - x_{vT^i}| \right].$$

Prova.

O novo custo de separação fracionário para a aresta $e = \{u, v\}$ é $w_e \sum_{T'<T} \ell_{T'} |\overline{x}_{uT'} - \overline{x}_{vT'}|$ e se u e v são ambos associados a T^i então a soma pode ser escrita como

$$\begin{aligned}
\sum_{T'<T^i} \ell_{T'} |\overline{x}_{uT'} - \overline{x}_{vT'}| &= \sum_{T'<T^i} \ell_{T'} \left| \frac{x_{uT'}}{x_{uT^i}} - \frac{x_{vT'}}{x_{vT^i}} \right| \\
&= \sum_{T'<T^i} \ell_{T'} \left| \left(\frac{x_{uT'}}{x_{uT^i}} - \frac{x_{vT'}}{x_{uT^i}} \right) - \left(\frac{x_{vT'}}{x_{vT^i}} - \frac{x_{vT'}}{x_{uT^i}} \right) \right| \\
&\leq \sum_{T'<T^i} \ell_{T'} \frac{1}{x_{uT^i}} |x_{uT'} - x_{vT'}| + \sum_{T'<T^i} \left[\frac{1}{x_{uT^i}} - \frac{1}{x_{vT^i}} \right] \ell_{T'} x_{vT'} \qquad (3.2) \\
&\leq \frac{1}{x_{uT^i}} \left[\sum_{T'<T^i} \ell_{T'} |x_{uT'} - x_{vT'}| \right] + \frac{1}{r-1} \ell_{T^i} x_{vT^i} \left[\frac{1}{x_{uT^i}} - \frac{1}{x_{vT^i}} \right] \qquad (3.3) \\
&= \frac{1}{x_{uT^i}} \left[\sum_{T'<T^i} \ell_{T'} |x_{uT'} - x_{vT'}| \right] + \frac{1}{r-1} \ell_{T^i} \left[\frac{x_{vT^i}}{x_{uT^i}} - 1 \right] \\
&= \frac{1}{x_{uT^i}} \left[\sum_{T'<T^i} \ell_{T'} |x_{uT'} - x_{vT'}| \right] + \frac{1}{r-1} \ell_{T^i} \left[\frac{x_{vT^i} - x_{uT^i}}{x_{uT^i}} \right] \\
&= \frac{1}{x_{uT^i}} \left[\sum_{T'<T^i} \ell_{T'} |x_{uT'} - x_{vT'}| + \frac{1}{r-1} \ell_{T^i} (x_{vT^i} - x_{uT^i}) \right],
\end{aligned}$$

onde (3.2) é obtida usando a hipótese que $x_{uT^i} \leq x_{vT^i}$ e (3.3) é obtida usando o lema 3.1.9.

□

Teorema 3.1.11 *Seja x uma solução do programa linear (KTT) aplicado a uma instância do problema ML, com o espaço métrico do conjunto de classes L aproximado através de uma árvore r-HST com $r > 2$, cujas folhas são classes. Seja c_{LP} e w_{LP} o custo de associação e separação, respectivamente, resultante da classificação fracionária x. O algoritmo Alg_ML encontra uma classificação com custo de associação esperado de c_{LP} e custo de separação esperado de no máximo*

$$(2 + \frac{4}{r-2}) w_{LP}.$$

Prova.

Ambas as afirmações sobre o custo da classificação são provadas por indução na profundidade da árvore.

Na base da indução a profundidade da árvore é 1, ou seja, a árvore T possui t filhos, que são as classes, e os valores $x_{uT^i} = x_{ui}$. Pelo teorema 3.1.3 o valor esperado para o custo de associação é c_{LP} e o valor esperado para o custo de separação é $2w_{LP}$.

Sejam $T^1, \ldots, T^t$ as subárvores que são filhas da raiz da árvore r-HST.

Primeira afirmação: "O algoritmo *Alg_ML* encontra uma classificação com custo de associação esperado de c_{LP}." Considere uma subárvore T^j no primeiro nível. Pelo lema 3.1.8, a probabilidade de um objeto u ser associado a subárvore T^j é x_{uT^j} e se u for associado a subárvore T^j então a probabilidade de associação, que pelo lema 3.1.1 é x_{ui}, é normalizada para

$$\overline{x}_{ui} = \frac{1}{x_{uT^j}} x_{ui},$$

para cada classe $i \in L(T^j)$. Por indução, uma vez que um objeto u é associado a uma subárvore T^j, este é atribuído a uma classe $i \in L(T^j)$ com probabilidade $\overline{x}_{ui}$ e assim a probabilidade do objeto u ser associado à classe i em $L(T^j)$ é $x_{uT^j}\overline{x}_{ui} = x_{ui}$. Logo, o custo de associação esperado é $\sum_{u \in P, i \in L} c_{ui} x_{ui}$, que é exatamente c_{LP}.

Segunda afirmação: "O algoritmo *Alg_ML* encontra uma classificação com custo de separação esperado de no máximo $(2 + \frac{4}{r-2}) w_{LP}$." O custo de separação fracionário para uma aresta $e = \{u, v\}$ é

$$w_e \sum_{T' < T} \ell_{T'} |x_{uT'} - x_{vT'}|.$$

O custo de separação é computado considerando o evento $\mathcal{E}'$ que indica se os objetos u e v são separados no nível mais alto e o evento $\mathcal{E}_i$ que indica que os objetos u e v são associados à mesma subárvore T^i. Seja $h(u, v)$ uma variável aleatória que indica a distância métrica entre as classes associadas aos objetos u e v. O custo de separação da aresta $e = \{u, v\}$ é $w_e h(u, v)$. Então o custo de separação esperado é $w_e E[h(u, v)]$ e a distância esperada entre as classes associadas aos objetos u e v pode ser escrita da seguinte maneira

$$E[h(u,v)] = E[h(u,v)|\mathcal{E}'] \cdot \Pr[\mathcal{E}'] + \sum_{i=1}^{t} E[h(u,v)|\mathcal{E}_i] \cdot \Pr[\mathcal{E}_i].$$

Pelo lema 3.1.8 temos que a probabilidade $\Pr[\mathcal{E}']$ de que dois objetos relacionados através de uma aresta sejam separados no primeiro nível é no máximo $\sum_{i=1}^{t} |x_{uT^i} - x_{vT^i}|$. O custo de separação neste caso é limitado pelo diâmetro da árvore que, usando propriedades das árvores r-HST, é no máximo $[2r/(r-1)]\ell_{T^i}$ para qualquer i. Assim, o primeiro termo da soma dada anteriormente é limitado por

$$E[h(u,v)|\mathcal{E}'] \cdot \Pr[\mathcal{E}'] \leq \frac{2r}{r-1} \sum_{i=1}^{t} \ell_{T^i} |x_{uT^i} - x_{vT^i}|. \tag{3.4}$$

Se ambos objetos forem associados à mesma subárvore T^i então, por indução no nível da subárvore e usando o resultado da equação (3.4), o custo de separação esperado é no máximo $2 + 4/(r-2)$ vezes o custo de separação fracionário da solução escolhida. A probabilidade

de que ambos objetos sejam associados à mesma subárvore T^i é no máximo $\min(x_{uT^i}, x_{vT^i})$ e usando o lema 3.1.10 podemos limitar o valor de $E[h(u,v)|\mathcal{E}_i] \cdot \Pr[\mathcal{E}_i]$ da seguinte forma

$$E[h(u,v)|\mathcal{E}_i] \cdot \Pr[\mathcal{E}_i] \leq \frac{2r}{r-2}\left[\sum_{T<T^i} \ell_T |x_{uT} - x_{vT}| + \frac{1}{r-1}\ell_{T^i}|x_{uT^i} - x_{vT^i}|\right].$$

Assim o limitante final para $E[h(u,v)]$ fica da seguinte forma

$$\begin{aligned} E[h(u,v)] &\leq \frac{2r}{r-1}\sum_{i=1}^{t} \ell_{T^i}|x_{uT^i} - x_{vT^i}| \\ &+ \frac{2r}{r-2}\sum_{i=1}^{t}\left[\sum_{T<T^i} \ell_T|x_{uT} - x_{vT}| + \frac{1}{r-1}\ell_{T^i}|x_{uT^i} - x_{vT^i}|\right] \\ &\leq \frac{2r}{r-2}\sum_{T'<T} \ell_{T'}|x_{uT'} - x_{vT'}|. \end{aligned}$$

Por definição, $w_{LP} = \sum_{T'<T} \ell_{T'}|x_{uT'} - x_{vT'}|$ o que conclui a prova do teorema 3.1.11. □

O fator de aproximação demonstrado no teorema 3.1.11 não leva em consideração o fato de que a árvore r-HST utilizada é uma aproximação do espaço métrico do conjunto de classes L. Para obter o fator de aproximação real precisamos aplicar o resultado de Bartal na aproximação probabilística de espaços métricos.

Segundo Bartal [2], para toda métrica d, há um conjunto $\mathcal{S}$ de árvores métricas r hierarquicamente bem separadas (r-hierarchically-well-separated tree metrics), onde cada $T \in \mathcal{S}$ tem uma função de distâncias d_T e uma distribuição de probabilidade p_T, tal que $d_T(u,v) \geq d(u,v)$ para todo $T \in \mathcal{S}$ e todo par de objetos u, v. Além disso, $d_T(u,v)$ não é muito maior que $d(u,v)$, mais precisamente, $\sum_{T\in\mathcal{S}} p_T d_T(u,v) \leq O(r \log m \log\log m) d(u,v)$. Utilizando este resultado Kleinberg e Tardos derivaram um algoritmo $O(\log m \log\log m)$-aproximado para o problema de classificação métrica em um espaço métrico geral.

O algoritmo *Alg_ML* tem como parâmetro de entrada uma árvore T que é uma árvore r-HST com a característica de que todas as classes são folhas. Antes de mostrar o fator de aproximação de $O(\log m \log\log m)$ para o algoritmo *Alg_ML*, vamos mostrar que podemos transformar uma árvore r-HST em uma r-HST onde todas as classes são folhas, com um acréscimo de um fator $2 + 1/r$ na distância.

Lema 3.1.12 *Seja T uma árvore r-HST com função de distância d_T, então existe uma árvore T' r-HST com função de distância $d_{T'}$ onde todas as classes são folhas, resultante da modificação da árvore T e para quaisquer duas classes i e j temos*

$$d_T(i,j) \leq d_{T'}(i,j) \leq (2 + 1/r) d_T(i,j).$$

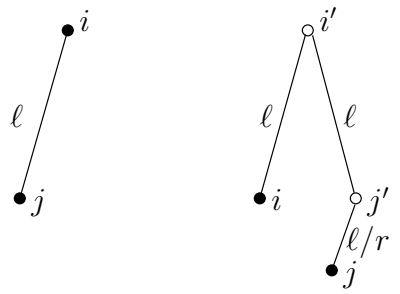

Figura 3.12: Modificação em uma árvore r-HST para que todas as classes sejam folhas.

Prova.

Para cada nó interno i da árvore T que é uma classe, substitua este nó i por um novo nó i' e coloque i como filho de i'. A maior alteração na distância ocorre se um nó i possui um filho j e ambos são classes, este caso esta ilustrado na figura 3.12. Seja $\ell = d_T(i,j)$. Então i passa a ser filho de i' e a distância entre i e i' é igual a ℓ, e j passa a ser filho de um novo nó j' e j' é filho de i'. Se a nova árvore T' é uma r-HST então a nova distância $d_{T'}(j,j') \leq \ell/r$, e temos que $d_{T'}(i,j) = d_{T'}(i,i') + d_{T'}(i',j') + d_{T'}(j',j) \leq 2\ell + \ell/r = (2+1/r)d_T(i,j)$.

□

Esta modificação das árvores r-HST é executada devido à Kleinberg e Tardos [17] considerarem que um nó interno de uma árvore r-HST pode ser uma classe. No entanto, se a árvore r-HST for gerada pelos algoritmos de Bartal [2, 3] ou Fakcheroenphol *et al.*[9] estas alterações não são necessárias, já que na árvore r-HST gerada por estes algoritmos as classes são as folhas e os nós internos representam um subgrafo da partição métrica hierárquica, veja a subseção 3.1.2.

A seguir, mostraremos um algoritmo que fornece um fator de aproximação esperado de $O(\log m \log\log m)$ para o problema ML, onde o espaço métrico é aproximado probabilisticamente usando uma árvore T, 3-HST, e em seguida é executado o algoritmo *Alg_ML* sobre esta árvore. Utilizando o algoritmo apresentado por Bartal [2] a árvore T é escolhida com probabilidade p_T em um conjunto $\mathcal{S}$.

ALGORITMO *Alg_ML'* (P, L, d, c, w)

onde (P, L, d, c, w) é uma instância para o problema ML.

1. Seja T a árvore devolvida pelo algoritmo de Bartal

2. Executa *Alg_ML* (P, L, d, c, w, T)

Figura 3.13: Algoritmo de Kleinberg e Tardos para o problema de classificação métrica.

Teorema 3.1.13 *O algoritmo Alg_ML', da figura 3.13, é um algoritmo $O(\log m \log\log m)$ aproximado que consome tempo polinomial para o problema da classificação métrica, onde m é o número de classes da instância de entrada.*

Prova.

Seja d uma função de distância métrica sobre o conjunto das classes L. Através da aproximação por árvores métricas de Bartal e do lema 3.1.12 encontramos um conjunto de árvores métricas 3-hierarquicamente-bem-separada que aproxima d. Denotamos por $\mathcal{S}$ este conjunto das árvores métricas e p_T, para todo $T \in \mathcal{S}$, a distribuição de probabilidade das árvores, tal que para cada par de classes $i, j \in L$ e cada árvore $T \in \mathcal{S}$, temos que $d_{ij} \leq d_T(i,j)$ e $\sum_{T\in\mathcal{S}} p_T d_T(i,j) \leq O(\log m \log\log m) d_{ij}$.

Seja $\mathcal{O}$ o custo da classificação ótima $f_{\mathcal{O}}$ para a métrica d. Seja $\mathcal{O}_T$, para uma árvore $T \in \mathcal{S}$, o custo da classificação $f_{\mathcal{O}}$ para as classes representadas pela árvore T. Por definição da aproximação através de árvores métricas temos que $\sum_{T\in\mathcal{S}} p_T \mathcal{O}_T$ é no máximo $O(\log m \log\log m)\mathcal{O}$.

Seja $\mathcal{A}$ uma variável aleatória que denota o custo resultante da utilização da métrica d na classificação devolvida pelo algoritmo *Alg_ML*, e $\mathcal{A}_T$ uma variável aleatória que indica o custo resultante da utilização da métrica d_T na classificação devolvida pelo algoritmo *Alg_ML*. Pela definição de aproximação através de árvores métricas temos que $\mathcal{A} \leq \mathcal{A}_T$, para todo $T \in \mathcal{S}$.

Lembre-se que o custo esperado da classificação resultante $E[\mathcal{A}]$ é no máximo $O(\log m \log\log m)\mathcal{O}$. Seja ε_T uma variável que corresponde ao evento que a árvore T é selecionada no algoritmo *Alg_ML'*. Temos que $\Pr(\varepsilon_T) = p_T$ e que $E[\mathcal{A}_T|\varepsilon_T] \leq 6\mathcal{O}_T$. O fator multiplicativo 6 é resultado da substituição do valor de r por 3 no teorema 3.1.11. Usando estas esperanças temos o seguinte.

$$\begin{aligned} E[\mathcal{A}] &= \textstyle\sum_{T\in\mathcal{S}} E[\mathcal{A}|\varepsilon_T] \cdot \Pr[\varepsilon_T] \\ &\leq \textstyle\sum_{T\in\mathcal{S}} p_T E[\mathcal{A}_T|\varepsilon_T] \\ &\leq 6 \textstyle\sum_{T\in\mathcal{S}} p_T \mathcal{O}_T \\ &= O(\log m \log\log m)\mathcal{O}. \end{aligned} \tag{3.5}$$

□

Note que o fator de aproximação de $O(\log m \log\log m)$ para o caso métrico geral ocorre devido à distorção de $O(\log m \log\log m)$ gerada ao aproximarmos o espaço métrico de L em uma árvore HST. Recentemente Fakcheroenphol *et al.* [9] apresentaram um novo limitante de $O(\log m)$ para a distorção causada ao aproximar um espaço métrico de m pontos através de árvores HST. Baseado no resultado de Fakcheroenphol *et al.* podemos alterar o enunciado do teorema 3.1.13 e obtemos seguinte teorema.

Teorema 3.1.14 *O algoritmo Alg_ML', da figura 3.13, é um algoritmo $O(\log m)$ aproximado que consome tempo polinomial para o problema da classificação métrica, onde m é o número de classes.*

3.2 Programação Linear com Interpretação de Fluxo em Rede

Chekuri *et al.* [7], motivados pelo fato de que para utilizar a formulação apresentada por Kleinberg e Tardos [17], é preciso uma aproximação do espaço métrico, desenvolveram uma nova formulação, que pode ser aplicada diretamente às instâncias do caso uniforme e do caso métrico.

Para definir o novo programa linear, Chekuri *et al.* [7] usaram duas variáveis:

- x_{ui}, como na formulação de Kleinberg e Tardos [17], é uma variável que indica que o objeto u está associado à classe i, para cada objeto $u \in P$ e cada classe $i \in L$,
- x_{uivj} indica que o objeto u está associado à classe i e o objeto v está associado à classe j, ou seja, que os objetos u e v são associados a classes diferentes.

Nesta nova formulação é mantida a restrição de que cada objeto deve ser associado a exatamente uma classe, o que é representado pela restrição

$$\sum_{i \in L} x_{ui} = 1.$$

Além disso, é acrescentada a restrição

$$\sum_{j=1}^{m} x_{uivj} - x_{ui} = 0$$

que força consistência nas variáveis x_{uivj}. Observe que x_{uivj} será 1 somente se $x_{ui} = 1$ e $x_{vj} = 1$. Já a restrição $x_{uivj} - x_{vjui} = 0$ expressa a simetria entre estas variáveis. Nesta formulação usaremos a notação $N(u)$ para especificar o conjunto de objetos que têm relação com o objeto u no grafo $G = (P, E)$. A formulação relaxada é apresentada a seguir.

$$\begin{array}{lll}
\min & \displaystyle\sum_{u \in P}\sum_{i \in L} c_{ui} \cdot x_{ui} + \sum_{e=\{u,v\} \in E}\sum_{i \in L}\sum_{j \in L} w_e \cdot d_{ij} \cdot x_{uivj} & \\
s.a. & \sum_{i \in L} x_{ui} = 1 & \forall\, u \in P, \\
 & \sum_{j \in L} x_{uivj} - x_{ui} = 0 & \forall\, u \in P, \forall\, v \in N(u), \forall\, i \in L, \quad \text{(CKNZ)} \\
 & x_{uivj} - x_{vjui} = 0 & \forall\, e = \{u, v\} \in E, \forall\, i, j \in L, \\
 & x_{ui} \geq 0 & \forall\, u \in P, \forall\, i \in L, \\
 & x_{uivj} \geq 0 & \forall\, e = \{u, v\} \in E, \forall\, i, j \in L.
\end{array}$$

Para auxiliar na compreensão e na análise desta formulação, Chekuri *et al.* [7] constroem uma rede de fluxo, levando em conta que o conjunto de classes $L = \{1, \ldots, m\}$ é representado por uma seqüência de inteiros. A rede de fluxos, ilustrada na figura 3.14, é construída da forma como segue.

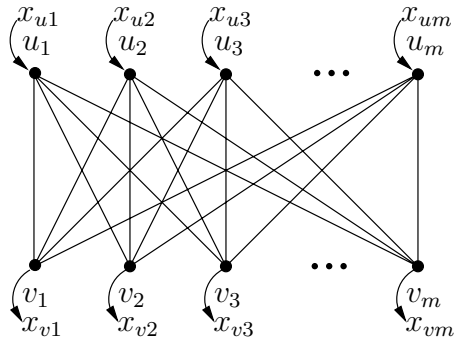

Figura 3.14: Rede de fluxos $H(u, v)$ para uma aresta $\{u, v\} \in E$.

Para cada aresta $\{u, v\} \in E$ é associado um grafo completo bipartido $H(u, v)$ com os vértices $\{u_1, \ldots, u_m\}$ e $\{v_1, \ldots, v_m\}$, estes vértices representam todas as possíveis classificações de u e v. Para cada par de vértices u_i, v_j, com $1 \leq i, j \leq m$, há uma aresta $\{u_i, v_j\}$ no grafo $H(u, v)$. Durante esta seção vamos nos referir às arestas $\{u_i, v_j\}$ por *links*.

Supondo que os valores das variáveis x_{ui}, para todo $u \in P$ e $i \in L$, foram determinados, usando estes valores o grafo $H(u, v)$ é interpretado como uma rede de fluxo da seguinte forma:

- a alimentação dos vértices u_i é x_{ui}, $\forall\ 1 \leq i \leq m$.
- a demanda dos vértices v_j é x_{vj}, $\forall\ 1 \leq j \leq m$.
- o custo de transportar uma unidade de fluxo de u_i para v_j é d_{ij}, $\forall\ 1 \leq i, j \leq m$.

Seja x uma solução do programa linear (CKNZ) e seja $e = \{u, v\} \in E$ uma aresta. Como as variáveis x_{ui} e x_{vi} são a produção e a demanda na rede de fluxos $H(u, v)$, podemos interpretar o valor atribuído à variável x_{uivj} como a quantidade de fluxo que passa pelo *link* $\{u_i, v_j\}$ em $H(u, v)$. A contribuição da aresta e para o custo de separação é o valor do transporte ótimo do fluxo entre $\{u_1, \ldots, u_m\}$ e $\{v_1, \ldots, v_m\}$.

No restante desta seção, vamos denotar por $d_{LP}(u, v)$ a distância entre as classes associadas aos objetos u e v em uma solução ótima (fracionária) do programa linear (CKNZ). Esta distância é o custo do transporte ótimo em $H(u, v)$, que é $\sum_{i,j \in L} d_{ij} x_{uivj}$.

Os custos do transporte ótimo nas redes $H(u, v)$, para todo $u, v \in P$, induz uma métrica no grafo $G = (P, E)$, já que, para qualquer $u, v, y \in P$, o custo do transporte de u para v não pode ser maior que a soma dos custos dos transportes de u para y e de y para v. Esta métrica é estudada na literatura de processamento de imagens com o nome de *earth mover's metric* [20, 21].

Esta rede de fluxo não é implementada realmente, devido ao seu alto custo. Ela serve apenas para auxiliar na compreensão e análise dos algoritmos.

Chekuri *et al.* [7] analisam a utilização do programa linear (CKNZ) para os problemas de classificação métrica uniforme, classificação métrica linear e classificação métrica truncada.

3.2.1 O caso da métrica uniforme

Kleinberg e Tardos com sua formulação (KTR) obtiveram uma 2-aproximação para este caso usando o algoritmo de arredondamento *Alg_UML*. Aplicando este algoritmo na solução fracionária do programa linear (CKNZ), Chekuri *et al.* obtiveram o mesmo fator de aproximação para o problema de classificação uniforme.

Para obter tal fator de aproximação, estes autores mostraram que, usando o resultado da formulação (CKNZ) no algoritmo *Alg_UML* apresentado por Kleinberg e Tardos [17], o valor da solução devolvida é menor ou igual ao valor da solução do algoritmo *Alg_UML*, que utiliza o resultado da formulação (KTR). Em outras palavras, a formulação (CKNZ) é semelhante a formulação (KTR) para o problema de classificação métrica uniforme.

Teorema 3.2.1 *O fator de aproximação do algoritmo Alg_UML não é maior que 2 quando usamos a formulação (CKNZ) no lugar da formulação (KTR).*

Prova.

Sabemos que utilizando a formulação (KTR) o algoritmo *Alg_UML* tem um fator de aproximação igual a 2, para que este fator não se altere é necessário mostrar que o valor devolvido pela formulação (CKNZ) não é maior que o valor obtido da formulação (KTR), quando aplicado a uma instância do problema de classificação métrica uniforme. A função objetivo da formulação (KTR) é

$$\sum_{u\in P}\sum_{i\in L} c_{ui}x_{ui} + \frac{1}{2}\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m} w_e|x_{ui} - x_{vi}|$$

enquanto que a função objetivo da formulação (CKNZ) é

$$\sum_{u\in P}\sum_{i\in L} c_{ui}x_{ui} + \sum_{e=\{u,v\}\in E}\sum_{i=1}^{m}\sum_{j=i}^{m} w_e d_{ij} x_{uivj}.$$

Note que o custo de atribuição, em ambas as formulações, é definido pelas variáveis x_{ui} e têm as mesmas contribuições para a função objetivo. Já a expressão que representa o custo de separação, que no programa linear (KTR) é dado por

$$\frac{1}{2}\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m} w_e|x_{ui} - x_{vi}|$$

e no novo programa linear (CKNZ) é dado por

$$\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m}\sum_{j=i}^{m} w_e d_{ij} x_{uivj},$$

são diferentes. Para que seja verdadeira a afirmação de que o fator de aproximação do algoritmo *Alg_UML* não é maior que 2, quando a formulação (CKNZ) é usada, é preciso mostrar que

$$\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m}\sum_{j=i}^{m} w_e d_{ij} x_{uivj} \leq \frac{1}{2}\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m} w_e |x_{ui} - x_{vi}|.$$

Como estamos analisando a desigualdade apenas para o caso de distâncias métricas uniformes, o valor assumido pela função de distância, utilizado no lado esquerdo da desigualdade, será 0 ou 1, sendo 0 somente quando $i = j$. Assim, se considerarmos apenas $i \neq j$ podemos eliminar a função de distância, e também, podemos dividir a desigualdade por w_e, eliminando o peso w_e. Fazendo estas alterações, temos:

$$\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m} x_{uivj} \;\leq\; \frac{1}{2}\sum_{e=\{u,v\}\in E}\sum_{i=1}^{m} |x_{ui} - x_{vi}|. \tag{3.6}$$

Esta desigualdade considera todas as arestas entre objetos e todas as classes. Vamos provar que para cada aresta do somatório acima a desigualdade é válida. Assim basta provar a seguinte desigualdade para cada aresta $e = \{u, v\} \in E$.

$$\sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m} x_{uivj} \;\leq\; \frac{1}{2}\sum_{i=1}^{m} |x_{ui} - x_{vi}|. \tag{3.7}$$

Claramente se a desigualdade (3.7) for válida então a desigualdade (3.6) também é válida.

Ao considerarmos uma aresta $e = \{u, v\}$ temos duas possibilidades. A primeira em que os objetos u e v possuem probabilidades de associação iguais, ou seja $x_{ui} = x_{vi}$, para todo $i \in L$, fazendo com que $\frac{1}{2}\sum_{i=1}^{m} |x_{ui} - x_{vi}| = 0$.

Neste caso, na formulação (CKNZ), pelas restrições

$$\sum_{j=1}^{m} x_{uivj} = x_{ui} \quad \forall\, u \in P, \forall\, v \in N(u), \forall\, i \in L$$

e $x_{uivj} - x_{vjui} = 0 \quad \forall\, e = \{u, v\} \in E, \forall\, i, j \in L$, as variáveis x_{uivj}, para $i \neq j$, são forçadas a assumir valor zero, enquanto que as variáveis x_{uivi} tendem a assumir valores diferentes de zero.

Para melhor ilustrar esta atribuição de valores para as variáveis x_{uivj}, considere a rede de fluxos $H(u, v)$. O custo de transportar uma unidade de fluxo do vértice u_i para o vértice v_j

é igual a $d(i,j)$ e a produção x_{ui} do vértice u_i é igual a demanda x_{vi} do vértice v_i, para todo $1 \leq i \leq m$. Baseado nisso, o transporte ótimo do fluxo entre os vértices u_i e v_i é feito através das arestas $\{u_i, v_i\}$. Estas arestas são escolhidas devido ao custo de transportar uma unidade de fluxo através delas ser igual a $d(i,i) = 0$. Podemos interpretar o valor da variável x_{uivj} como a quantidade de fluxo que passa pela aresta $\{u_i, v_j\}$. Assim, devido à probabilidade de associação dos objetos u e v serem iguais, as variáveis x_{uivj} que assumem valores diferentes de zero são aquelas em que $i = j$. Como estas variáveis não são consideradas no somatório, temos que $\sum_{i=1}^{m} \sum_{j=1, j\neq i}^{m} x_{uivj} = 0$.

A segunda possibilidade é de que os objetos u e v possuem probabilidades de associação diferentes, fazendo com que $\frac{1}{2} \sum_{i=1}^{m} |x_{ui} - x_{vi}| > 0$. Neste caso, na rede de fluxos $H(u,v)$, o transporte do fluxo que é considerado para um vértice u_i é apenas aquele que não é consumido, ou produzido por v_i, ou seja, $|x_{ui} - x_{vi}|$. Assim o fluxo total na rede $H(u,v)$, que é representado por $\sum_{i=1}^{m} \sum_{j=1}^{m} x_{uivj}$, é igual a $\sum_{i=1}^{m} \max\{0, x_{ui} - x_{vi}\}$. Esta somatória representa o fluxo enviado do conjunto de vértices u_i para o conjunto de vértices v_i e que pela restrição $\sum_{i=1}^{m} x_{ui} = 1$ do programa linear (CKNZ) é igual ao fluxo no sentido contrário. Assim podemos reescrever esta somatória como $\frac{1}{2} \sum_{i=1}^{m} |x_{ui} - x_{vi}|$, ou seja, ela é igual à contribuição da formulação (KTR). □

3.2.2 O caso da métrica linear

Para o problema de classificação métrica linear (LML) Chekuri *et al.* [7] apresentam um algoritmo que, utilizando uma solução fracionária do programa linear (CKNZ), obtém uma solução com valor esperado igual à solução fracionária do programa linear (CKNZ). No problema de classificação métrica linear temos as mesmas características de um problema de classificação métrica, com a diferença que para cada classe é atribuído um número inteiro positivo não repetido e a função de distância entre as classes é dada pelo módulo da diferença entre elas; i.e., $d : L \times L \rightarrow \mathbb{N}^+$, $d_{ij} = |i - j|$.

O Algoritmo para o Problema LML

O algoritmo para o problema de classificação métrica linear, apresentado na figura 3.15, tem como ponto de partida a resolução do programa linear (CKNZ), em seguida é escolhido um valor aleatório θ entre 0 e 1, que é utilizado no arredondamento da solução fracionária do programa linear (CKNZ).

Para entender melhor o procedimento de arredondamento, veja a figura 3.16. Nesta figura temos os valores α de um objeto u em relação a todas as classes do conjunto $L = \{1, 2, 3, 4\}$. Observe que o objeto u tem maior probabilidade de ser associado à classe 3, devido ao intervalo entre $\alpha(u, 2)$ e $\alpha(u, 3)$ ser o maior intervalo em $[0, 1]$, e o tamanho deste intervalo é

ALGORITMO ALG_LML (P, L, d, c, w)
onde (P, L, d, c, w) é uma instância para o problema LML.
1. Seja x uma solução para o programa linear (CKNZ)
2. $\theta \leftarrow random[0, 1]$
3. Para todo $u \in P$
4. Para $i = 1$ até m
5. $\alpha(u, i) \leftarrow \sum_{j=1}^{i} x_{uj}$
6. Para todo $u \in V$
7. Seja i tal que $\alpha(u, i-1) < \theta \leq \alpha(u, i)$
8. $f(u) \leftarrow i$
9. Devolva f

Figura 3.15: Algoritmo para o arredondamento do resultado do programa linear (CKNZ).

x_{u3}. A garantia de que todos os objetos serão associados a alguma classe é dada pela restrição $\sum_{i=1}^{m} x_{ui} = 1$.

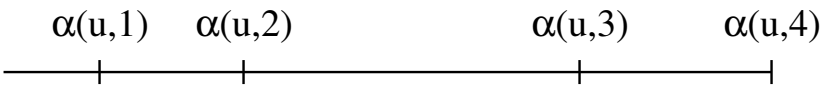

Figura 3.16: Exemplo de valores para α.

O valor esperado da solução inteira devolvida pelo algoritmo *Alg_LML*, para o problema de classificação métrica linear, é igual ao valor da solução fracionária devolvido pelo programa linear (CKNZ). Para mostrar que isso é verdade temos que analisar a esperança do custo de associação e separação. Para cada objeto $u \in P$ definimos uma variável aleatória $L(u)$, onde $L(u)$ representa a classe a quem o objeto u é associado no algoritmo *Alg_LML*, e a probabilidade de $L(u) = i$ é x_{ui}. Isto significa que o custo de associação esperado para um objeto u é $\sum_{i=1}^{m} x_{ui} c_{ui}$ que é exatamente o custo de associação no programa linear (CKNZ).

Lema 3.2.2 *O valor esperado da distância entre as classes associadas aos objetos u e v, para uma aresta $\{u, v\} \in E$, é*

$$E[d(L(u), L(v))] = \sum_{i=1}^{m} |\alpha(u, i) - \alpha(v, i)|.$$

Prova.

Para auxiliar na análise, definimos as variáveis binárias $Z_1, \ldots, Z_{m-1}$ cujos valores são

definidos da seguinte forma:

$$Z_i = \begin{cases} 1 & \text{se } \min\{L(u), L(v)\} \leq i < \max\{L(u), L(v)\} \\ 0 & \text{caso contrário.} \end{cases}$$

Em outras palavras Z_i é 1 se i estiver no intervalo entre $L(u)$ e $L(v)$. Então, como $d_{ij} = |i - j|$, fica claro que

$$d(L(u), L(v)) = \sum_{i=1}^{m-1} Z_i.$$

Conseqüentemente, $E[d(L(u), L(v))] = \sum_{i=1}^{m-1} E[Z_i]$. Afirmamos que

$$E[Z_i] = \Pr[Z_i = 1] = |\alpha(u, i) - \alpha(v, i)|. \tag{3.8}$$

O lema é facilmente provado usando a igualdade 3.8. Para provar a igualdade 3.8 suponha, sem perda de generalidade, que $\alpha(u, i) \geq \alpha(v, i)$. Se $\theta < \alpha(v, i)$ então $L(u)$ e $L(v)$ são menores ou iguais a i. Se $\theta > \alpha(u, i)$ então $L(u)$ e $L(v)$ são maiores que i. Em ambos os casos $Z_i = 0$. Se $\theta \in [\alpha(v, i), \alpha(u, i)]$ então $L(u) \leq i$ e $L(v) > i$, o que implica que $Z_i = 1$. Assim, $\Pr[Z_i = 1]$ é exatamente $|\alpha(u, i) - \alpha(v, i)|$.

□

De posse da esperança da distância entre as classes atribuídas aos extremos de uma aresta $e = \{u, v\}$, podemos estimar qual é a contribuição desta aresta para a função objetivo. Como indicado na seção 3.2, a contribuição da aresta $e = \{u, v\} \in E$ para a função objetivo é igual ao custo do transporte ótimo do fluxo no grafo completo bipartido $H(u, v)$. Vale lembrar que o valor da distância entre as classes associadas aos objetos u, v no programa linear (CKNZ) é $d_{LP}(u, v) = \sum_{i,j} d_{ij} \cdot x_{uivj}$.

Lema 3.2.3 *A seguinte desigualdade é válida para a métrica linear*

$$d_{LP}(u, v) \geq \sum_{i=1}^{m} |\alpha(u, i) - \alpha(v, i)|.$$

Prova.

Analisando a estrutura da rede $H(u, v)$ podemos observar que: Se a produção de u_i, que é igual a x_{ui}, for igual ao consumo de v_i, que é igual a x_{vi}, para todo i, significa que os dois objetos u e v serão associados à mesma classe i. Devido ao custo de transportar uma unidade de fluxo de u_i para v_j ser igual a d_{ij}, quando x_{ui} é igual a x_{vi}, para todo i, o custo de transportar o fluxo na rede $H(u, v)$ é zero, ou seja, o custo de separação é zero.

No caso em que a produção x_{ui} é maior que o consumo x_{vi} temos um fluxo excedente em relação à classe i, neste caso de $x_{ui} - x_{vi}$. Este fluxo excedente será consumido por um outro nó e para tanto precisa ser "transportado" através da rede $H(u, v)$. Vale lembrar que o custo

de transportar uma unidade de fluxo de u_i para v_j é igual a d_{ij}. Se x_{vi} for maior que x_{ui}, a quantidade de fluxo excedente é $x_{vi} - x_{ui}$. Assim, o fluxo excedente na rede de fluxos $H(u, v)$ em relação a uma classe i é $|x_{ui} - x_{vi}|$.

O que é consumido no intervalo $\{1, 2, \ldots, i\}$ é $\min\{\alpha(u, i), \alpha(v, i)\}$ e por conseqüência $|\alpha(u, i) - \alpha(v, i)|$ é a quantidade de fluxo excedente, que será enviada para fora deste intervalo, o que significa que será cobrada para que esta quantia excedente de fluxo seja transportada.

Aplicando este argumento a todos os valores de i, obtemos que o custo de transporte ótimo do fluxo em $H(u, v)$ é precisamente $\sum_{i=1}^{m} |\alpha(u, i) - \alpha(v, i)|$.

□

Assim obtemos que o custo esperado da aresta $e = \{u, v\} \in E$, após o arredondamento, não é maior que sua contribuição para a função objetivo do programa linear, ou seja, o *gap* de integralidade do programa linear (CKNZ) para o caso da métrica linear é um.

3.2.3 Melhora no fator de aproximação para o caso de distâncias métricas lineares truncadas (TML)

No capítulo 4 é apresentado uma abordagem para o problema de classificação métrica desenvolvida por Gupta e Tardos [11], que utiliza algoritmos de fluxo em redes para obter uma solução para uma instância do problema de classificação métrica truncada. O problema de classificação métrica truncada é uma variação do problema de classificação métrica onde as classes são definidas como inteiros de 1 até m e a distância entre duas classes i e j é dada pela função de distância $d_{ij} = \min(|i - j|, M)$, onde M é o valor máximo da distância.

Gupta e Tardos, utilizando algoritmos de fluxo em redes, obtiveram um algoritmo polinomial que obtém uma solução de valor no máximo 4 vezes o valor ótimo para o problema TML. Com o programa linear (CKNZ) Chekuri *et al.* [7] obtiveram uma $(2 + \sqrt{2})$-aproximação, melhorando o resultado prévio de Gupta e Tardos [11].

A métrica truncada do conjunto de classes L pode ser vista como um grafo formado por um caminho contendo m vértices, que representam as m classes, e para cada par de vértices i, j, onde $d_{ij} > M$, é acrescentada uma aresta de tamanho M, assim como ilustrado na figura 3.17. Podemos considerar este grafo como um caminho com o truncamento em M implícito. Isto permite utilizar, com modificações apropriadas, a idéia utilizada para o caso das distâncias métricas lineares.

Ao contrário do algoritmo apresentado por Gupta e Tardos [11] para o caso das distâncias métricas truncadas, o algoritmo apresentado por Chekuri *et al.* [7] utiliza uma solução do programa linear. O algoritmo *Alg_TML_CKNZ*, apresentado na figura 3.18, tem como parâmetros de entrada o conjunto de objetos P, o conjunto de classes L e um valor M' que é maior ou igual a M e é definido na análise do algoritmo.

O algoritmo *Alg_TML_CKNZ* generaliza o algoritmo de arredondamento para o problema

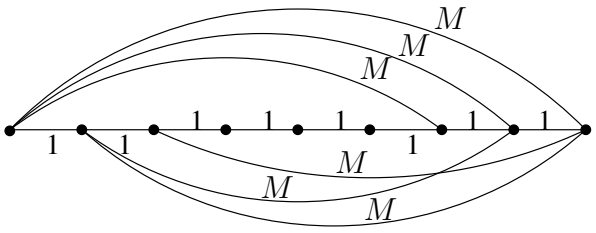

Figura 3.17: Grafo para a métrica truncada com $M = 5$.

ALGORITMO *Alg_TML_CKNZ* (P, L, d, c, w, M')
onde (P, L, d, c, w) é uma instância para o problema TML.

1. Seja x uma solução para o programa linear (CKNZ)
2. Enquanto existir objeto não classificado
3. Escolha aleatoriamente um inteiro ℓ em $[-M' + 2, m]$ com probabilidade uniforme
4. Seja I_ℓ o intervalo $[\ell, \ell + M' - 1]$
5. $\theta \leftarrow random[0, 1]$
6. Para todo $u \in P$ não está classificado
7. Para todo $i \in I_\ell$
8. Se $\sum_{j=\ell}^{i-1} x_{uj} < \theta \leq \sum_{j=\ell}^{i} x_{uj}$
9. $f(u) \leftarrow i$
10. Devolva f

Figura 3.18: Algoritmo para o caso das distâncias truncadas utilizando a formulação (CKNZ).

UML e LML de uma forma natural. Uma vez que ℓ é selecionado e o intervalo I_ℓ é definido, o arredondamento é similar ao da métrica linear. A principal diferença entre o algoritmo *Alg_LML* e o algoritmo *Alg_TML_CKNZ* é que no primeiro um objeto sempre é associado a uma classe em uma iteração. Já no algoritmo *Alg_TML_CKNZ* um objeto pode não ser associado em uma iteração, devido a cada iteração considerar apenas parte do conjunto de classes. Se dois objetos são classificados em iterações separadas assumimos que a distância entre estas classes é M.

A melhora no fator de aproximação é obtida através de uma análise cuidadosa do algoritmo. A análise guia para a escolha de um M' que resulta na melhor garantia de aproximação, onde $M' \geq M$.

Lema 3.2.4 *Em uma iteração qualquer, a probabilidade de um objeto u, não classificado, ser associado à classe i naquela iteração é exatamente*

$$x_{ui} \cdot M'/(m + M' - 1).$$

A probabilidade de u ser associado a alguma classe é $M'/(m + M' - 1)$. Assim

$$\Pr[L(u) = i] = x_{ui}.$$

Prova.

Se ℓ é escolhido no primeiro passo de uma iteração e se $i \in I_\ell$, então a probabilidade de associar o objeto u à classe i é exatamente x_{ui} e, se $i \notin I_\ell$, a probabilidade é zero. O número de intervalos que contêm i é M', devido ao fato de podermos escolher ℓ para delimitar o intervalo que contem i de M' formas diferentes. Assim, a probabilidade do objeto u ser associado a uma classe i em uma iteração é a probabilidade de associar o objeto u a classe i vezes a probabilidade de se escolher a classe i que resulta em $x_{ui} \cdot M'/(m + M' - 1)$.

□

Segue deste lema que, com alta probabilidade, todos os objetos são associados em $O(m \log n)$ iterações. O próximo lema delimita o valor esperado da distância entre $L(u)$ e $L(v)$ em função de M, relembrando que $d_{LP} = \sum_{e=\{u,v\}\in E} \sum_{i\in L} \sum_{j\in L} d_{ij} \cdot x_{uivj}$.

Lema 3.2.5 *A distância esperada entre $L(u)$ e $L(v)$ satisfaz a seguinte desigualdade* $E[d(L(u), L(v))] \leq (2 + \max\{\frac{2M}{M'}, \frac{M'}{M}\})d_{LP}(u, v)$.

O lema 3.2.5 será provado mais adiante devido à necessidade de outros resultados anteriores.

Teorema 3.2.6 *Há um algoritmo $(2+\sqrt{2})$-aproximado para o problema de classificação métrica quando as distâncias são métricas lineares truncadas.*

Prova.

Note que o algoritmo e a análise são facilmente generalizados para o caso em que M' é um número real. Observando a desigualdade do lema 3.2.5, o valor de M' que minimiza a função $\max\{\frac{2M}{M'}, \frac{M'}{M}\}$ é $\sqrt{2}M$. Assim, pelos lemas 3.2.4 e 3.2.5, se $M' = \sqrt{2}M$ temos um algoritmo $(2 + \sqrt{2})$-aproximado para o problema de classificação métrica quando as distâncias são métricas lineares truncadas.

□

Com o objetivo de provar o lema 3.2.5 vamos analisar o efeito do procedimento de arredondamento na distância esperada entre as classes associadas aos objetos de uma aresta $e = \{u, v\} \in E$. Consideraremos uma iteração onde u e/ou v serão associados a uma classe. Podem acontecer dois casos:

- Somente um dos objetos é associado em uma iteração, ou seja, são separados, e pagaremos a distância M.
- Os dois objetos são associados na mesma iteração e pagaremos a distância entre as classes que estão no mesmo intervalo.

O interesse principal aqui é na distância esperada entre as classes associadas a u e v em uma iteração. Vamos utilizar o grafo $H(u, v)$, definido no início da seção 3.2, para nos auxiliar na prova do lema 3.2.5. Denotaremos as arestas $\{u_i, v_j\}$ do grafo $H(u, v)$ por *links*.

Dado um intervalo $I_\ell = [\ell, \ell + M' - 1]$, podemos dividir os *links* interessantes de H em 3 categorias. Consideramos *links* interessantes aqueles que têm pelo menos um dos extremos no intervalo I_ℓ. As 3 categorias de *links* interessantes estão ilustradas na figura 3.19 e são divididas da seguinte forma:

- *Links* de cruzamento à esquerda, que ligam um vértice u_i, com $i \in I_\ell$, a um vértice v_j, tal que $j < \ell$ e ℓ é o limite à esquerda do intervalo $I_\ell = [\ell, \ell + M' - 1]$. O conjunto de tais *links* é denotado por *LCRS*(I_ℓ). Observe que o *link* de cruzamento à esquerda também pode ser $\{u_j, v_i\}$, ou seja, o vértice u_j é quem está à esquerda do intervalo I_ℓ.
- *Links* internos são os *links* que ligam u_i a v_j, onde $i, j \in I_\ell$ e formam o conjunto *INT*(I_ℓ).
- *Links* de cruzamento à direita são aqueles que ligam um vértice u_i, com $i \in I_\ell$, a um vértice v_j, tal que $j > \ell + M' - 1$, onde $\ell + M' - 1$ é o limite à direita do intervalo $I_\ell = [\ell, \ell + M' - 1]$. Tais *links* formam o conjunto *RCRS*(I_ℓ). Observe que o *link* de cruzamento a direita também pode ser $\{u_j, v_i\}$, ou seja, o vértice u_j é quem está à direita do intervalo I_ℓ.
- Denotamos por *CRS*(I_ℓ) a união de *LCRS*(I_ℓ) com *RCRS*(I_ℓ).

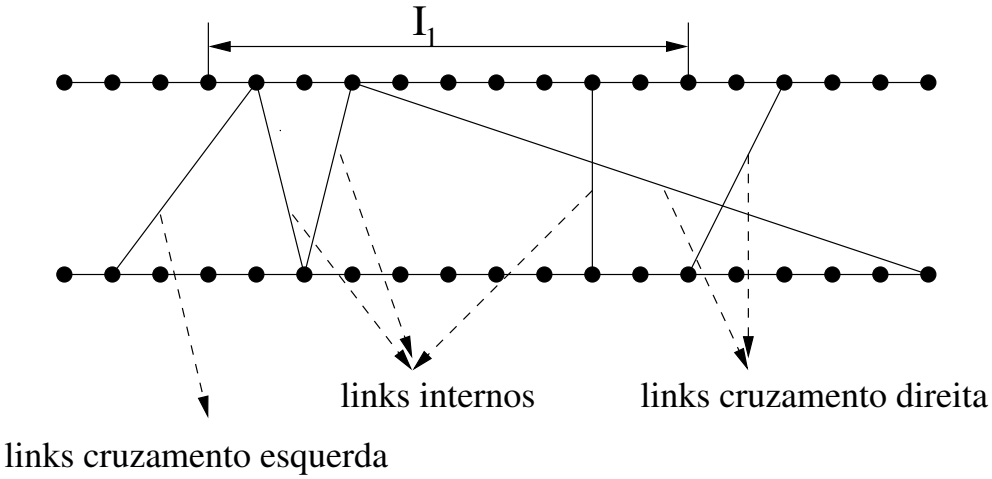

Figura 3.19: *Links* interessantes para o intervalo I_ℓ

Usaremos a seguinte notação no resto da prova.

- $e = \{u_i, v_j\}$ é um *link* (aresta) em $H(u, v)$ que liga os vértices u_i e v_j.
- $d_e = d_{ij}$ é a distância truncada entre as classes i e j.
- $|e| = |i - j|$ é a distância linear entre as classes i e j.

- x_e representa o valor de x_{uivj}, que é interpretado como a quantidade de fluxo que passa do vértice u_i para o vértice v_j.
- Seja $e \in \mathit{CRS}(I_\ell)$ e seja $i \in I_\ell$ a classe a qual e está ligada. Denotamos por e_ℓ a quantidade $(\ell + M' - 1 - i)$, que é a distância de i até o limite direito do intervalo I_ℓ.
- $x(u, I_\ell) = \sum_{i \in I_\ell} x_{ui}$ é o fluxo de u associado pelo programa linear (CKNZ) ao intervalo I_ℓ.

Lema 3.2.7 *A probabilidade de u e v serem separados em uma iteração, ou seja, apenas um dos objetos u e v é associado a uma classe pertencente ao intervalo I_ℓ, dado que ℓ é escolhido no primeiro passo da iteração, é no máximo*

$$\sum_{e \in \mathit{CRS}(I_\ell)} x_e.$$

Prova.

Olhando pela perspectiva de fluxo na rede $H(u, v)$, a quantidade de fluxo que é produzida no intervalo I_ℓ e não é consumida pelos vértices internos ao intervalo I_ℓ é dada por $|x(u, I_\ell) - x(v, I_\ell)|$, que é exatamente a probabilidade de separação dos objetos u e v. Esta quantidade excedente de fluxo é transportada para fora do intervalo I_ℓ através dos *links* de cruzamento à esquerda e à direita. Assim

$$|x(u, I_\ell) - x(v, I_\ell)| \leq \sum_{e \in \mathit{CRS}(I_\ell)} x_e. \tag{3.9}$$

□

Lema 3.2.8 *Para dois vértices u e v não classificados antes de uma iteração, seja p_ℓ a distância esperada entre as classes associadas a estes objetos, condicionado ao evento que ℓ é escolhido no primeiro passo da iteração e ambos foram associados a classes pertencentes ao intervalo I_ℓ. Então:*

$$p_\ell \leq \sum_{e \in \mathit{CRS}(I_\ell)} e_\ell x_e + \sum_{e \in INT(I_\ell)} |e| x_e.$$

Algumas intuições antes da prova. Uma vez que ℓ é escolhido, o arredondamento é exatamente o mesmo que para o caso de métricas lineares, restrito ao intervalo I_ℓ. Vimos no lema 3.2.2 o valor exato do custo esperado para o processo de arredondamento. No entanto não temos um lema equivalente ao lema 3.2.3 que delimita a distância utilizada pelo programa linear, devido a I_ℓ ser apenas um subintervalo do caminho formado pelas classes de $1, \ldots, m$, e ter seu tamanho truncado.

A principal dificuldade é em relação aos *links* que pertencem ao conjunto *CRS*(I_ℓ). Estes *links* são "carregados" de um valor e_ℓ (diferente de d_e, que é o valor pago pelo programa linear) para que possamos pagar o custo do transporte ótimo do fluxo induzido pelos valores fracionários das variáveis x_{ui} e x_{vi}.

Prova.

Fixe ℓ e, sem perda de generalidade, suponha que $x(u, I_\ell) \geq x(v, I_\ell)$. Com probabilidade $q = x(v, I_\ell)$ ambos, u e v são associados a classes em I_ℓ. Analisando a distância esperada baseada no evento em que ambas as classes associadas aos objetos u e v pertencem ao intervalo I_ℓ, uma vez que o intervalo I_ℓ é fixado, o arredondamento é muito similar ao caso das métricas lineares.

Seja

$$\alpha(u, i) = \sum_{j=\ell}^{\ell+i} x_{uj},$$

para $0 \leq i \leq M'$, o fluxo acumulado de u nas primeiras $i + 1$ classes do intervalo I_ℓ. No caso em que u e v são associados a classes no intervalo I_ℓ assumimos que a distância é linear e ignoramos o truncamento. Pelo Lema 3.2.3 a distância esperada entre u e v é igual a $\sum_{i=0}^{M'-1} |\min\{q, \alpha(u, i)\} - \alpha(v, i)|$ que limitamos superiormente por $\sum_{i=0}^{M'-1} |\alpha(u, i) - \alpha(v, i)|$. Observe que a contribuição de $\alpha(u, i)$ para a distância, quando $\alpha(u, i)$ é maior que q, é desconsiderada por ser a quantidade de fluxo que não é consumida pelos vértices internos ao intervalo I_ℓ na rede $H(u, v)$. Baseado nesse limitante podemos substituir p_ℓ por $\sum_{i=0}^{M'-1} |\alpha(u, i) - \alpha(v, i)|$. A seguinte desigualdade é válida

$$\sum_{i=0}^{M'-1} |\alpha(u, i) - \alpha(v, i)| \leq \sum_{e \in \mathit{CRS}(I_\ell)} e_\ell x_e + \sum_{e \in \mathit{INT}(I_\ell)} |e| x_e. \tag{3.10}$$

Para provar que a equação (3.10) é verdadeira, consideramos cada *link* $e = (u_a, v_b) \in$ *CRS*$(I_\ell) \bigcup$ *INT*(I_ℓ) e somamos sua contribuição para o termo $q_i = |\alpha(u, i) - \alpha(v, i)|$, com $0 \leq i < M'$. Seja $e = \{u_a, v_b\} \in$ *INT*(I_ℓ). É claro que e contribui exatamente com x_e para q_i se $a \leq i$ e $b > i$ ou $a > i$ e $b \leq i$. Caso contrário sua contribuição é 0. Assim, a contribuição total de e para $\sum q_i$ é $x_e|a - b| = x_e|e|$.

Para melhor compreender a contribuição do *link* $e = \{u_a, v_b\} \in$ *INT*(I_ℓ) para $\sum q_i$, observe a figura 3.20. A região sombreada da figura representa o fluxo computado para q_i. Além disso, podemos considerar o *link* $e = \{u_a, v_b\}$ como um *link* de cruzamento da região q_i, o que significa que é uma quantidade de fluxo x_e excedente da região q_i, e como o custo de se transportar uma unidade de fluxo pelo *link* $e = \{u_a, v_b\}$ é igual a $d(a, v) = |a - b|$ temos que a contribuição total de e para $\sum q_i$ é $x_e|a - b| = x_e|e|$.

Agora, suponha que $e = \{u_a, v_b\} \in$ *LCRS*(I_ℓ) e, sem perda de generalidade, que $a \geq \ell$ e $b < \ell$. O caso em que $a < \ell$ e $b \geq \ell$ é análogo. O *link* e contribui com x_e para $\alpha(u, i)$,

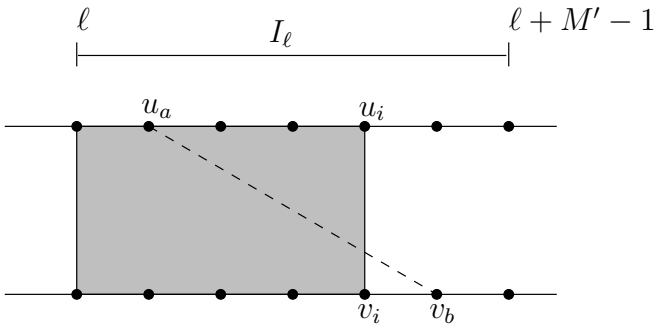

Figura 3.20: Contribuição do *link* $\{u_a, v_b\} \in$ *INT*(I_ℓ) para $q_i = |\alpha(u,i) - \alpha(v,i)|$.

com $a - \ell \leq i < M'$, e contribui com 0 para $\alpha(v,i)$, com $0 \leq i < M'$, já que b está fora do intervalo I_ℓ. Assim, a contribuição de e para q_i é x_e, se $a - \ell \leq i < M'$, e 0 caso contrario. A contribuição total de e para $\sum q_i$ é $x_e|\ell + M' - 1 - a| = e_\ell x_e$. Um argumento similar é valido quando $e \in$ *RCRS*(I_ℓ).

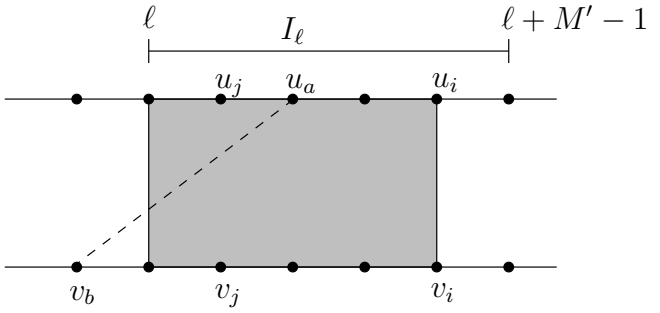

Figura 3.21: Contribuição do *link* $\{u_a, v_b\} \in$ *CRS*(I_ℓ) para $q_i = |\alpha(u,i) - \alpha(v,i)|$.

Para melhor compreender a contribuição do *link* $e = \{u_a, v_b\} \in$ *LCRS*(I_ℓ) para $\sum q_i$, observe a figura 3.21. Nesta figura, até o momento em que $i = 1$, que considera o fluxo até os vértices u_j e v_j, a contribuição do *link* $\{u_a, v_b\}$ para a quantidade q_i é nula. O valor x_e passará a contribuir para q_i no momento em que $i = a - \ell$, que é o momento que a quantidade x_{ua}, produzida pelo vértice u_a, é computada para $\alpha(u,i)$, e essa contribuição é mantida para todos os valores de i maiores que $a - \ell$, até o fim do intervalo I_ℓ. Como v_b, o outro extremo do *link* $\{u_a, v_b\}$, está fora do intervalo I_ℓ, a quantidade x_{vb} não influenciará no valor de $\alpha(v,i)$ e por conseqüência não influencia no valor de q_i fazendo com que a contribuição total de e para $\sum q_i$ é $x_e|\ell + M' - 1 - a| = e_\ell x_e$. A mesma argumentação é válida para $e \in$ *RCRS*(I_ℓ).

□

Prova do lema 3.2.5

Prova.

Dada uma iteração onde u e v são dois objetos não classificados, denotamos por $\Pr[u \wedge v]$

a probabilidade de ambos serem associados, $\Pr[u \oplus v]$ a probabilidade de exatamente um ser associado e $\Pr[u \vee v]$ a probabilidade de pelo menos um ser associado. Vamos definir um limite superior para $E[d(L(u), L(v))]$ da seguinte forma: Se u e v são separados em alguma iteração então, podemos simplificar o limite superior da distância por M. Usando esta simplificação, a seguinte desigualdade é válida.

$$E[d(L(u), L(v)] \leq \frac{\Pr[u \oplus v] \cdot M + \Pr[u \wedge v] \cdot E[d(L(u), L(v))|u \wedge v]}{\Pr[u \vee v]}. \tag{3.11}$$

Definimos um limite superior para os componentes da parte direita da desigualdade como segue:

- Limitamos inferiormente $\Pr[u \vee v]$ por $\Pr[u]$ que, pelo lema 3.2.4, é igual a $M'/(m + M' - 1)$.
- Limitamos superiormente $\Pr[u \oplus v]$ por $\frac{1}{(m+M'-1)} \sum_\ell \sum_{e \in \mathit{CRS}(I_\ell)} x_e$, que é a probabilidade de escolher um intervalo I_ℓ vezes a probabilidade de separar os objetos u e v que, pelo lema 3.2.7, é $\sum_\ell \sum_{e \in \mathit{CRS}(I_\ell)} x_e$.
- Pela definição de p_ℓ, no lema 3.2.8, temos o seguinte:

$$\Pr[u \wedge v] E[d(f(u), f(v))|u \wedge v] = \frac{1}{(m + M' - 1)} \sum_\ell p_\ell.$$

Substituindo estes valores na desigualdade (3.11) e utilizando o lema 3.2.8 para limitar p_ℓ, que é o valor esperado da distância entre as classes associadas a dois objetos u e v, considerando o evento em que ambos os objetos são associados a classes pertencentes ao mesmo intervalo I_ℓ, temos

$$\begin{aligned}
E[d(L(u), L(v)] &\leq \left[\frac{\left(\frac{1}{(m+M'-1)} \sum_\ell \sum_{e \in \mathit{CRS}(I_\ell)} x_e \right) M + \left(\frac{1}{(m+M'-1)} \sum_\ell p_\ell \right)}{\frac{M'}{(m+M'-1)}} \right] \\
&\leq \frac{1}{M'} \left[\sum_\ell \sum_{e \in \mathit{CRS}(I_\ell)} x_e M + \sum_\ell p_\ell \right] && (3.12) \\
&\leq \frac{1}{M'} \left[\sum_\ell \sum_{e \in \mathit{CRS}(I_\ell)} x_e M + \sum_\ell \left(\sum_{e \in \mathit{CRS}(I_\ell)} e_\ell x_e + \sum_{e \in \mathit{INT}(I_\ell)} |e| x_e \right) \right] && (3.13) \\
&\leq \frac{1}{M'} \sum_\ell \left(\sum_{e \in \mathit{CRS}(I_\ell)} (M + e_\ell) x_e + \sum_{e \in \mathit{INT}(I_\ell)} |e| x_e \right) && (3.14)
\end{aligned}$$

$$\leq \frac{1}{M'}\sum_e x_e\left(\sum_{\ell:e\in \textit{CRS}(I_\ell)}(M+e_\ell)+\sum_{\ell:e\in \textit{INT}(I_\ell)}|e|\right) \quad (3.15)$$
$$= \sum_e x_e\overline{d_e},$$

onde $\overline{d_e}=\frac{1}{M'}(\sum_{\ell:e\in \textit{CRS}(I_\ell)}(M+e_\ell)+\sum_{\ell:e\in \textit{INT}(I_\ell)}|e|)$. A passagem da equação (3.12) para a equação (3.13) ocorre devido ao lema 3.2.8. As equações (3.14) e (3.15) são equivalentes, dado que na equação (3.14) é considerada primeiro a escolha de ℓ e posteriormente computada a contribuição dos *links* para aquele intervalo. Já na equação (3.15) é considerado um *link* e computada qual é sua contribuição para todos os intervalos.

□

O lema 3.2.9 mostra que $\overline{d_e}$ é limitado superiormente por $(2+\max\{2M/M', M'/M\})d_e$. Isto conclui a prova do lema 3.2.5, levando em consideração que $d_{LP}(u,v)=\sum_e x_e d_e$.

Lema 3.2.9 $\overline{d_e}=\frac{1}{M'}(\sum_{\ell:e\in \textit{CRS}(I_\ell)}(M+e_\ell)+\sum_{\ell:e\in \textit{INT}(I_\ell)}|e|)\leq(2+\max\left\{\frac{2M}{M'},\frac{M'}{M}\right\})d_e.$

Prova.

Dado que $M'\geq M$ e $d_e\leq M\leq M'$, $\overline{d_e}$ é avaliado separadamente para três diferentes tipos de arestas, $|e|\geq M'$, $|e|<M$ e $M\leq|e|<M'$. Seja $e=\{u_i,v_j\}$ um *link* em $H(u,v)$ e, sem perda de generalidade, suponha que $i\leq j$. O outro caso é similar.

- $|e|\geq M'$: Neste caso está claro que e não pode ser uma aresta interna de intervalo, para qualquer intervalo I_ℓ, ou seja, a contribuição do *link* e para $\overline{d_e}$ é dada apenas pelo termo da somatória que computa os *links* de cruzamento. Também $d_e=M$. Assim

$$M'\overline{d_e} = \sum_{\ell:e\in \textit{CRS}(I_\ell)}(M+e_\ell) \quad (3.16)$$
$$= \sum_{\ell:e\in \textit{LCRS}(I_\ell)}(M+e_\ell)+\sum_{\ell:e\in \textit{RCRS}(I_\ell)}(M+e_\ell) \quad (3.17)$$
$$= \sum_{\ell:e\in \textit{LCRS}(I_\ell)}(M+M'+\ell-1-j)+\sum_{\ell:e\in \textit{RCRS}(I_\ell)}(M+M'+\ell-1-i) \quad (3.18)$$
$$= \sum_{\ell\geq j-M'}^{j}(M+M'+\ell-1-j)+\sum_{\ell\geq i-M'}^{i}(M+M'+\ell-1-i) \quad (3.19)$$
$$\leq \sum_{\ell=0}^{M'}(M+M'-\ell)+\sum_{\ell=0}^{M'}(M+M'-\ell) \quad (3.20)$$
$$\leq M'(2M+M')$$
$$= M'(2+M'/M)d_e.$$

O conjunto *CRS*(I_ℓ) dos *links* que cruzam um intervalo I_ℓ é a união de dois conjuntos, os *links LCRS*(I_ℓ) que cruzam o intervalo I_ℓ no lado esquerdo e os *links RCRS*(I_ℓ) que cruzam o intervalo I_ℓ no lado direito, justificando a passagem da equação (3.16) para (3.17). A quantidade e_ℓ, para uma aresta $e = \{u_i, v_j\}$ com $i \in I_\ell$, representa a distância do vértice u_i interno ao intervalo I_ℓ até o extremo direito do intervalo I_ℓ, que é igual a $M' + \ell - 1 - i$ e aplicando esse valor na equação (3.17) temos a equação (3.18).

Dado um *link* $e = \{u_i, v_j\}$ de cruzamento à esquerda, tal que $i \leq j$ e $|i - j| \geq M'$. As restrições da escolha de ℓ nos somatórios da equação (3.18) podem ser substituídas por $j - M' \leq \ell \leq j$ e $i - M' \leq \ell \leq i$, que são os valores de ℓ que mantém o *link* e como um *link* de cruzamento à esquerda e à direita, respectivamente, obtendo assim a equação (3.19).

Claramente o número de possíveis valores que ℓ pode assumir é M' em ambos somatórios e a diferença $\ell - j$, que é a distância do elemento interno ao intervalo I_ℓ ao limite direito do intervalo I_ℓ varia entre 0 e M', resultando assim na equação (3.20).

- $|e| < M$: Neste caso o *link* e pode tanto ser um *link* interno quanto um *link* de cruzamento e $d_e = |e|$.

$$
\begin{aligned}
M'\overline{d_e} &= \sum_{\ell : e \in \textit{CRS}(I_\ell)} (M + e_\ell) + \sum_{\ell : e \in \textit{INT}(I_\ell)} |e| \\
&= \sum_{\ell : e \in \textit{LCRS}(I_\ell)} (M + e_\ell) + \sum_{\ell : e \in \textit{RCRS}(I_\ell)} (M + e_\ell) + \sum_{\ell : e \in \textit{INT}(I_\ell)} |e| \quad (3.21) \\
&= \sum_{\ell > i}^{j} (M + M' + \ell - 1 - j) + \sum_{\ell > i - M'}^{j - M'} (M + M' + \ell - 1 - i) + \sum_{\ell \geq j - M'}^{i} |e| \quad (3.22) \\
&\leq \sum_{\ell=0}^{|j-i|} (M + M' - \ell) + \sum_{\ell=0}^{|j-i|} (M + \ell) + \sum_{\ell=0}^{M' - |j-i|} |e| \quad (3.23) \\
&= |e|M + |e|M' - |e|\frac{|e|}{2} + |e|M + |e|\frac{|e|}{2} + (M' - |e|)|e| \quad (3.24) \\
&= (2M' + 2M - |e|)|e| \\
&\leq (2M' + 2M)|e| \\
&= M'(2 + 2M/M')d_e.
\end{aligned}
$$

Como *CRS*(I_ℓ) = *LCRS*$(I_\ell) \cup$ *RCRS*(I_ℓ) temos a equação (3.21). Para que o *link* $e = \{u_i, v_j\}$ seja um *link* de cruzamento à esquerda, o valor de ℓ tem que ser maior que i e menor que j. Para e ser um *link* de cruzamento à direita ℓ tem que ser entre $i - M'$ e $j - M'$ e para e ser um *link* interno ℓ tem que estar entre $j - M'$ e i. Utilizando esses valores para ℓ e substituindo os valores de e_ℓ, na equação (3.21) temos (3.22).

No primeiro somatório da equação (3.22) temos $|j-i|$ possíveis valores para ℓ fazendo com que os termos do somatório variem de $M+M'-|j-i|$, para $\ell = i+1$, a $M+M'-0-1$, para $\ell = j$. No segundo somatório ℓ pode assumir $|j-i|$ valores diferentes fazendo com que os termos do somatório variem de $M+0$, para $\ell = i-M'+1$, a $M+|j-i|$, para $\ell = j-M'$. No terceiro somatório temos $M'-|j-i|$ possíveis valores para ℓ. Substituindo estes valores na equação (3.22) obtemos a equação (3.23). Como $|j-i| = |e|$, substituindo e simplificando os somatórios, temos a equação (3.24).

- $M \leq |e| < M'$: Neste caso o *link* e pode tanto ser um *link* interno quanto um *link* de cruzamento e $d_e = M$.

$$
\begin{aligned}
M'\overline{d_e} &= \sum_{\ell:e\in \mathit{CRS}(I_\ell)} (M+e_\ell) + \sum_{\ell:e\in \mathit{INT}(I_\ell)} |e| \\
&= \sum_{\ell:e\in \mathit{LCRS}(I_\ell)} (M+e_\ell) + \sum_{\ell:e\in \mathit{RCRS}(I_\ell)} (M+e_\ell) + \sum_{\ell:e\in \mathit{INT}(I_\ell)} |e| \\
&= \sum_{\ell>i}^{j} (M+M'+\ell-1-j) + \sum_{\ell>i-M'}^{j-M'} (M+M'+\ell-1-i) + \sum_{\ell\geq j-M'}^{i} |e| \\
&\leq \sum_{\ell=0}^{|j-i|} (M+M'-\ell) + \sum_{\ell=0}^{|j-i|} (M+\ell) + \sum_{\ell=0}^{M'-|j-i|} |e| \\
&= |e|M + |e|M' - |e|\frac{|e|}{2} + |e|M + |e|\frac{|e|}{2} + (M'-|e|)|e| \\
&= (2M'+2M-|e|)|e| \\
&\leq M'(M'+2M) \qquad (3.25) \\
&= M'(2+M'/M)d_e. \qquad (3.26)
\end{aligned}
$$

Observe que as igualdades e desigualdades anteriores a desigualdade (3.25) são iguais as desigualdades do caso onde $|e| < M$. Por conseqüência são válidas. Limitando $|e|$ por M' temos a equação (3.25) e como o valor de $d_e = M$ temos (3.26).

Assim, de acordo com o tamanho do *link*, o valor $\overline{d_e}$ pode ser $(2+M'/M)d_e$, se $e = \{u_i, v_j\}$ e $|i-j| \geq M$, ou $(2+2M/M')d_e$, se $e = \{u_i, v_j\}$ e $|i-j| < M$. Ou seja o valor de $\overline{d_e}$ é no máximo $(2+\max\left\{\frac{2M}{M'}, \frac{M'}{M}\right\})d_e$. Este valor nos guia para a escolha de $M' = \sqrt{2}M$, que é o valor que minimiza a função $\max\left\{\frac{2M}{M'}, \frac{M'}{M}\right\}$.

□

3.2.4 O caso de métricas gerais

Com esta nova formulação Chekuri *et al.* [7] conseguiram o mesmo fator de aproximação de $O(\log m)$ para o caso com métrica geral, apresentado anteriormente por Kleinberg e Tar-

dos [17], que utiliza da aproximação do espaço métrico do conjunto de classes L por árvores métricas r-HST e aplica o programa linear (KTT) sobre esta aproximação.

A nova formulação serve como uma alternativa na obtenção de uma solução para o problema de classificação com métricas gerais. Em contraste à abordagem dada por Kleinberg e Tardos, Chekuri *et al.* aplicam diretamente o programa linear (CKNZ) sobre uma instância do problema de classificação métrica.

A solução devolvida pelo programa linear (CKNZ) é usada para identificar uma aproximação métrica HST determinística do espaço métrico, de forma que o custo da solução fracionária devolvida pelo programa linear (CKNZ) nesta métrica HST seja no máximo $O(\log m)$ vezes o custo do programa linear no espaço métrico original. Isto pode ser feito utilizando o seguinte resultado de Charikar *et al.*[6], acrescido do resultado de Fakcheroenphol *et al.*[9].

Proposição 3.2.10 *Seja d uma métrica arbitrária definida sobre m pontos e seja α uma função não negativa definida sobre todos os pares de pontos. Então d pode ser deterministicamente aproximada por uma árvore métrica HST T com função de distância d_T tal que*

$$\sum_{i,j} \alpha(i,j) d_T(i,j) \leq O(\log m \log\log m) \sum_{i,j} \alpha(i,j) d_{ij}.$$

Dada uma solução do programa linear (CKNZ), aplicamos a proposição 3.2.10 com a função de peso

$$\alpha(i,j) = \sum_{e=\{u,v\} \in E} w_e x_{uivj},$$

para $0 \leq i,j \leq m$. Isto é, $\alpha(i,j)$ é o peso fracionário da aresta $e = \{u,v\}$, onde $f(u) = i$ e $f(v) = j$ ou $f(u) = j$ e $f(v) = i$. Como a árvore métrica HST resultante, d_T, altera apenas a métrica entre as classes e não a classificação fornecida pelo programa linear (CKNZ), a solução fracionária continua uma solução factível para esta nova métrica e tem um acréscimo no custo de no máximo $O(\log m)$. Assim, se arredondarmos a solução fracionária utilizando a métrica d_T, acrescentaremos um fator constante no custo da solução e obtém-se um algoritmo $O(\log m)$-aproximado.

Este arredondamento da solução do programa linear (CKNZ) pode ser feito utilizando o algoritmo *Alg_ML* de Kleinberg e Tardos, apresentado na figura 3.8, que apresenta um fator de aproximação de $O(1)$ para o problema de classificação métrica, utilizando uma árvore métrica HST para aproximar o espaço métrico das classes, resultando em um algoritmo $O(\log m)$-aproximado.

Capítulo 4

Algoritmo que Utiliza Fluxo em Redes

Os algoritmos apresentados no capítulo 3 tem como passo inicial a resolução de um programa linear. Neste capítulo apresentaremos uma abordagem desenvolvida por Gupta e Tardos [11], que não depende da resolução de nenhum programa linear para a obtenção de uma solução.

4.1 Fluxo em Redes e os Problemas de Classificação Métrica

Gupta e Tardos [11] apresentam um algoritmo com fator de aproximação constante para o problema de classificação métrica linear truncada, *Truncated Line Metric Labeling Problem (TML)*. No problema TML um conjunto de classes $L = \{1, \ldots, m\}$ é visto como um caminho de tamanho m onde as classes representam os vértices e as arestas $(i, i+1)$ entre as classes têm tamanho 1. Para cada par de classes $\{i, j\}$ com $|i-j| > M$ é adicionada uma aresta de tamanho M, resultando assim na métrica linear truncada. Assim a distância entre duas classes i e j é dada pelo menor valor entre a diferença, em módulo, entre as classes ou um valor máximo M, isto é, $d : L \times L \to \mathbb{N}^+$, $d(i, j) = \min\{|i - j|, M\}$. A métrica truncada pode ser vista como o grafo da figura 4.1.

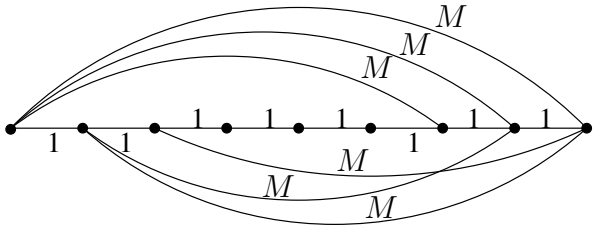

Figura 4.1: Grafo G_L para a métrica truncada do conjunto de classes L com $M = 5$.

O algoritmo apresentado por Gupta e Tardos [11] para o problema TML é um algoritmo de

busca local, e tem fator de aproximação igual a 4. A idéia principal é que o espaço métrico truncado pode ser aproximado por uma árvore estrela gerada aleatoriamente, onde as folhas são intervalos de tamanho no máximo M. Veja a figura 4.2.

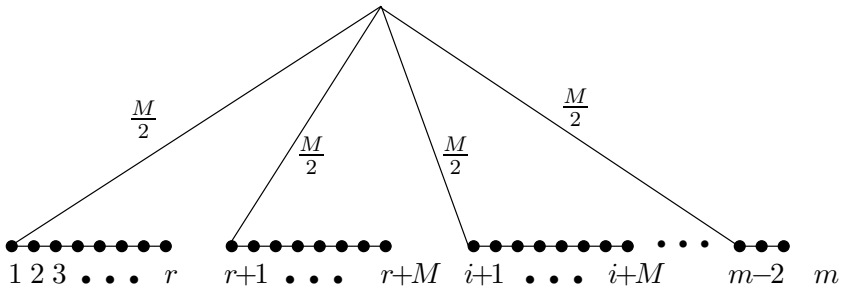

Figura 4.2: Árvore de distâncias truncadas, onde $i = r+M$.

Os passos utilizados para aproximar o grafo G_L da métrica truncada (veja a figura 4.1) por uma árvore aleatória T (veja a figura 4.2), são os seguintes:

- Escolha aleatoriamente um valor $r \in \{1, 2, \ldots, M\}$.
- Para cada posição $1 \leq k \leq m$, tal que $k - \lfloor \frac{k}{M} \rfloor M = r$, apague cada aresta $\{i, j\}$ do grafo G_L, onde $i \leq k$ e $j > k$ (veja a figura 4.2). Como resultado desta operação temos um conjunto de intervalos em função da escolha do valor r, que vamos denotar por S_r. Note que apenas o primeiro e o último intervalo podem ter comprimentos menores que M, onde o primeiro intervalo tem tamanho igual a r e o último intervalo tem tamanho igual a $(m - r) - \lfloor \frac{m-r}{M} \rfloor M$.
- Crie uma árvore com um nó s como sendo a raiz.
- Para cada intervalo I em S_r, conecte o menor vértice do intervalo I ao vértice s através de uma aresta de comprimento $M/2$.

Baseado na forma como é construída a árvore T que aproxima o espaço métrico do conjunto de classes L temos o seguinte fato.

Fato 4.1.1 *Para qualquer par de classes $i, j \in L$ a probabilidade que i e j pertençam a intervalos diferentes é igual a d_{ij}/M.*

Como conseqüência deste fato temos o seguinte teorema em relação à distância esperada entre duas classes na árvore T.

Teorema 4.1.2 *Para qualquer par de classes $i, j \in L$ a distância $d_T(i, j)$, em qualquer árvore T aleatória, é no mínimo d_{ij} e o valor esperado de $d_T(i, j)$ é $3d_{ij}$.*

Prova.

Se $d_{ij} < M$ e as classes i e j pertencerem ao mesmo intervalo após a escolha de r, então a distância entre i e j é igual a d_{ij}. No entanto, se i e j pertencem a intervalos diferentes, então a distância entre eles será dada pelo menor caminho entre eles, que é no mínimo M. Assim o valor de $d_T(i,j) \geq d_{ij}$, como anunciado.

Assumindo que as classes i e j são separadas, o valor esperado de $d_T(i,j)$ depende de quem vai ser a classe que representa o início do intervalo que contem j, vamos chamar esta classe que é a primeira classe que do intervalo que contem j de r^2, e vamos chamar de r^1 a primeira classe do intervalo que contem a classe i. Note que temos d_{ij} possíveis valores para r^2, que vai de $i+1$ até $i+d_{ij}$, isto pode ser visto melhor na figura 4.3.

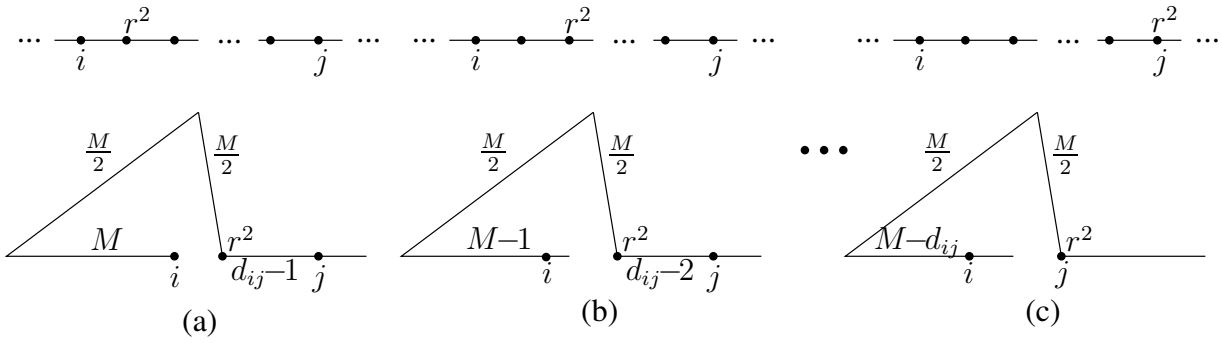

Figura 4.3: Possíveis valores de $d_T(i,j)$, quando i e j pertencem a intervalos diferentes.

Denotamos por ε o evento onde i e j são separados. O valor esperado de $d_T(i,j)$, considerando que i e j pertencem a intervalos diferentes, é baseado na probabilidade da escolha de r^2, que é dado por

$$\begin{aligned} E[d_T(i,j)|\varepsilon] &= \frac{1}{d_{ij}} \sum_{i=1}^{d_{ij}} 2M - (i-1) + (d_{ij} - i) \\ &= \frac{1}{d_{ij}} \sum_{i=1}^{d_{ij}} 2M - (i-1) + (i-1) \\ &= \frac{1}{d_{ij}} 2M d_{ij} \\ &= 2M. \end{aligned}$$

Assim, o valor esperado de $d_T(i,j)$, considerando que i e j pertencem a intervalos diferentes, é $2M$. Como a probabilidade de i e j pertencem a intervalos diferentes é $\frac{d_{ij}}{M}$, o valor esperado de $d_T(i,j)$ é dado por

$$\begin{aligned} E[d_T(i,j)] &= \frac{d_{ij}}{M}2M + \left(1 - \frac{d_{ij}}{M}\right) d_{ij} \\ &= 2d_{ij} + d_{ij} - \frac{d_{ij}^2}{M} \\ &\leq 3d(i,j). \end{aligned}$$

□

Veremos que no algoritmo, esta árvore não é criada diretamente, e sim uma partição S_r aleatória do espaço métrico do conjunto de classes em intervalos, como descrito anteriormente.

4.1.1 O Algoritmo para o Problema TML

O algoritmo apresentado por Gupta e Tardos, que denotamos por *Alg_TML_GT*, inicia com uma classificação arbitrária f_0, e iterativamente obtém uma nova classificação f' de menor custo através de movimentos locais. O movimento local aqui citado consiste em encontrar uma nova classificação através da interpretação de cortes em uma rede de fluxo, que é construída com base nos custos de associação, separação e as distâncias truncadas, que são aproximadas através das árvores de intervalos, citadas anteriormente. Dada uma classificação f, vamos denotar por $Q(f)$ o seu custo de classificação e Q_0 o custo da classificação inicial, ou seja, $Q_0 = Q(f_0)$.

ALGORITMO *Alg_TML_GT* (P, L, ϵ)

$P :=$ conjunto de objetos; $L :=$ conjunto de classes.

$\epsilon :=$ folga de aproximação.

1. Seja f_0 uma classificação aleatória

2. Repita $\frac{m+M}{M}(\log Q_0 + \log \epsilon^-)$ vezes

3. Escolha aleatoriamente um valor r entre $-M$ e m.

4. $I \leftarrow \{r+1, r+2, \ldots, r+M\} \cap \{1, \ldots, m\}$

5. Construa a rede de fluxos N_I associada ao intervalo I

6. Execute *MinCut*(N_I)

7. Seja f' a classificação resultante do corte na rede de fluxos (N_I)

8. Se $Q(f') < Q(f_0)$ então

9. $f_0 \leftarrow f'$

10. Devolva f_0

Figura 4.4: Algoritmo para o caso das distâncias truncadas de Gupta e Tardos.

Na figura 4.4 apresentamos o algoritmo *Alg_TML_GT* para o problema de classificação métrica linear truncada. O algoritmo começa sua execução com uma classificação arbitrária

f_0 e a cada iteração do algoritmo é escolhido aleatoriamente um intervalo I, criada uma rede de fluxos N_I, que é associada ao intervalo I, e então é aplicado o algoritmo do corte mínimo. Baseado nesse corte é escolhida uma nova classificação para os objetos.

A probabilidade de se escolher um r que dá origem a uma árvore aleatória com um conjunto de intervalos S_r é igual a $1/M$. Dado um conjunto de intervalos S_r, a probabilidade de se escolher um intervalo I' em S_r é igual a $1/|S_r|$, que é igual a M/m. Assim a probabilidade de um intervalo I' ser escolhido é igual a $1/m$. Note que a probabilidade de escolher um intervalo I, nos passos 3 e 4 do algoritmo *Alg_TML_GT*, não é maior que a probabilidade de escolher um intervalo I' através da escolha de uma árvore aleatória e então, escolher o intervalo no conjunto de intervalos desta árvore.

Antes de fazer a análise do algoritmo *Alg_TML_GT* vamos ver como é construída a rede de fluxos N_I e como é obtida uma nova classificação através de um corte mínimo nesta rede.

A rede de fluxos N_I é um grafo orientado e sua construção é baseada na relação entre os objetos e nos custos de associação. Para construir a rede de fluxo, são consideradas as classes pertencentes a um intervalo I. Vamos assumir que estas classes são $\{i+1, i+2, \ldots, j\}$, onde o número de classes é dado por $|i-j|$, que é no máximo M. A seguir descrevemos a forma como é construída a rede de fluxo N_I associada ao intervalo I.

- Acrescente um vértice s como a origem do fluxo e t como o destino do fluxo.
- Para cada objeto $u \in P$ adicione $|i-j|$ nós da forma $u_{i+1}, \ldots, u_j$.
- Acrescente arestas direcionadas (u_k, u_{k+1}) com capacidade igual a c_{uk}, para todo $i+1 \leq k \leq j$, onde $u_{j+1} = t$.
- Acrescente arestas direcionadas (u_{k+1}, u_k) com capacidade igual a infinito, para todo $i+1 \leq k \leq j$.
- Acrescente aresta direcionada (s, u_{i+1}) com capacidade $D(u)$ que é definido como: se $f_0(u) \in I$ então $D(u) = \infty$, senão $D(u) = c(u, f_0(u))$, ou seja, a aresta (s, u_{i+1}) tem capacidade igual a infinito, se a classe a quem o objeto u está associado, na atual classificação, pertencer ao intervalo I, ou a capacidade da aresta (s, u_{i+1}) será igual ao custo da classificação atual do objeto u se $f_0(u) \notin I$.
- Acrescente uma aresta direcionada (u_{i+1}, s) com capacidade igual a infinito.

A figura 4.5 ilustra uma parte da rede, que se assemelha a uma "corrente", que é construída para todos os objetos $u \in P$. Os vértices s e t são criados apenas uma vez, ou seja, todas as "correntes" são ligadas a estes dois vértices.

Para ver a razão dessa construção, considere qualquer corte mínimo $s-t$ em N_I. A inclusão das arestas (u_{k+1}, u_k), para $i+1 \leq k \leq j$ e (u_{i+1}, s) com pesos igual a infinito, na rede de

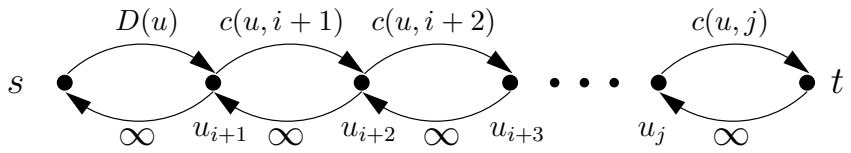

Figura 4.5: A "corrente" para o vértice u.

fluxos N_I, faz com que um corte $s-t$, de capacidade mínima, inclua apenas uma aresta, que pode ser a aresta (s, u_{i+1}) ou uma das arestas (u_k, u_{k+1}). Se a aresta que pertencer ao corte mínimo $s-t$ for (u_k, u_{k+1}), este corte corresponde à associação do objeto u com a classe k e o seu valor é igual a c_{uk}. Já se a aresta (s, u_{i+1}) pertencer ao corte, significa que a classificação anterior deste objeto será mantida. Note que o valor do corte mínimo $s-t$ na rede de fluxos N_I representa o custo de associação.

Para que o custo de separação seja acrescentado no valor do corte mínimo da rede N_I, são acrescentadas arestas entre as correntes da seguinte forma:

a) Para cada aresta $e = \{u, v\}$ no grafo original são adicionadas arestas orientadas (u_k, v_k) e (v_k, u_k) cada uma com capacidade w_e, onde $i + 1 < k \leq j$.

b) Se, na classificação atual, $f_0(u) \in I$ e $f_0(v) \in I$, para $e = \{u, v\}$, então é adicionada uma aresta (u_{i+1}, v_{i+1}) com capacidade w_e.

c) Se, na classificação atual, $f_0(u) \notin I$ e $f_0(v) \in I$, para $e = \{u, v\}$, então é adicionada uma aresta (u_{i+1}, v_{i+1}) com capacidade $w_e d(f_0(u), i + 1)$.

d) Se, na classificação atual, $f_0(u) \notin I$ e $f_0(v) \notin I$, para $e = \{u, v\}$, então é adicionado um novo vértice v_{uv} que é conectado à rede através das arestas (u_{i+1}, v_{uv}) e (v_{uv}, u_{i+1}), ambas com capacidade $w_e d(f(u), i + 1)$, (v_{uv}, v_{i+1}) e (v_{i+1}, v_{uv}), ambas com capacidade $w_e d(f(v), i + 1)$ e uma aresta (s, v_{uv}) com capacidade $w_e d(f_0(u), f_0(v))$.

A figura 4.6 ilustra a rede de fluxos N_I resultante, para o caso onde $f_0(u) \notin I$ e $f_0(v) \notin I$. Nesta figura as arestas com pesos igual a infinito foram omitidas.

Um corte $s-t$ de capacidade mínima efetuado na rede N_I, com as novas arestas, representa, além do custo de associação, o custo de separação. Podemos ter quatro tipos de cortes na rede de fluxos N_I em relação a dois objetos u e v. Vamos nos referenciar a estes quatro tipos de cortes por C_1, C_2, C_3 e C_4, assim como ilustrado na figura 4.7. Seguem as características de cada tipo de corte e os seus significados.

- Corte do tipo C_1: como podemos ver na figura 4.7(a), as arestas que fazem parte deste corte são (u_k, u_{k+1}) e (v_k, v_{k+1}) que significa que os objetos u e v são associados à mesma classe k e, por conseqüência, o custo de separação é igual a zero.

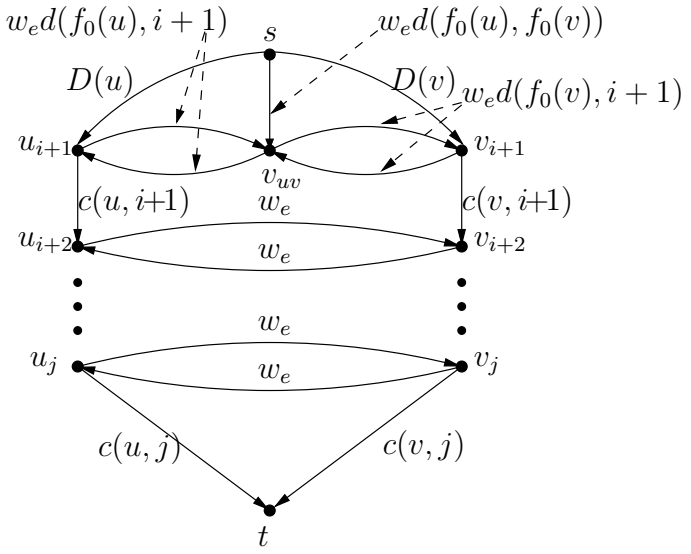

Figura 4.6: Rede N_I após o acréscimo das arestas que representam o custo de separação.

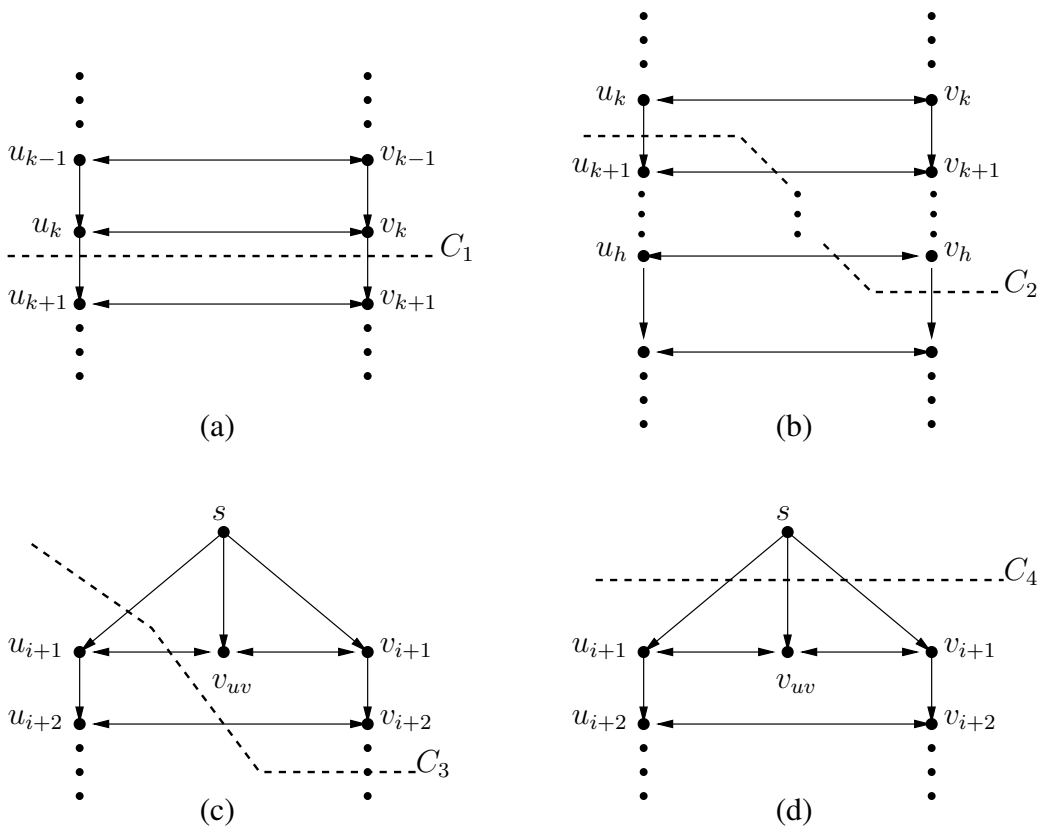

Figura 4.7: Exemplos de cortes na rede N_I.

- Corte do tipo C_2: neste caso, as arestas que fazem parte do corte são (u_k, u_{k+1}), (v_h, v_{h+1}) e (v_l, u_l), para $k < l \leq h$, conforme mostra a figura 4.7(b). Como visto anteriormente, as arestas (u_k, u_{k+1}) e (v_h, v_{h+1}) representam a classificação com $f'(u) = k$ e $f'(v) = h$ com a contribuição para a capacidade do corte de c_{uk} e c_{vh}. Já as arestas (v_l, u_l), para $k < l \leq h$, representam o custo de separação $w_e d(f'(u), f'(v)) = w_e|k - h|$.

- Corte do tipo C_3: neste tipo de corte, um dos objetos é associado a uma classe pertencente ao intervalo I e o outro mantém sua classificação anterior. No caso da figura 4.7(c), o corte C_3 indica que o objeto u mantém sua associação anterior, onde $f_0(u) \notin I$, com o custo $D(u)$, que neste caso é $c(u, f_0(u))$, enquanto que o objeto v é associado a uma classe no intervalo I. O custo de separação neste caso é $w_e d(f_0(u), i + 1)$, dado pela aresta (v_{uv}, u_{i+1}), mais $w_e|i + 1 - k|$, dado pelas arestas (v_l, u_l), para $i + 1 < l \leq k$, supondo que a aresta (v_k, v_{k+1}) pertença ao corte C_3.

- Corte do tipo C_4: como podemos ver na figura 4.7(d), este tipo de corte é composto pelas arestas (s, u_{i+1}), (s, v_{uv}) e (s, v_{i+1}), significando que os objetos u e v mantiveram sua classificação anterior juntamente com seus custos de associação e separação, que neste caso é dado por $w_e d(f_0(u), f_0(v))$, pela aresta (s, v_{uv}), mais $c(u, f_0(u))$ e $c(v, f_0(v))$, pelas arestas (s, v_{uv}) e (s, v_{i+1}).

Note que um corte mínimo na rede de fluxos N_I corta no máximo uma das arestas entre os vértices (v_{uv}, u_{i+1}), (v_{uv}, v_{i+1}) e (s, v_{uv}). Isto ocorre devido à capacidade de (s, v_{uv}) ser maior que a soma das capacidades das arestas (v_{uv}, u_{i+1}) e (v_{uv}, v_{i+1}), ou seja, $w_e d(f_0(u), f_0(v)) \leq w_e d(f_0(u), i + 1) + w_e d(i + i, f_0(v))$, o que é verdade devido a função de distância no conjunto de classes ser métrica e respeitar a desigualdade triangular.

Antes de formalizar a relação entre cortes da rede N_I e a reclassificação, devemos considerar dois casos extremos: (1) quando o intervalo I consiste de todas as classes e (2) quando o intervalo I tem uma única classe, $I = \{i + 1\}$. No caso (1) a construção é a mesma que a apresentada anteriormente e segundo Boykov *et al.*[4] e Ishikawa e Geiger [13], um corte mínimo na rede de fluxos associada ao intervalo I, que contém todas as classes, equivale à classificação ótima para a métrica linear em I. No caso (2) a construção da rede continua a mesma e o corte indicará quais serão os objetos reclassificados para a classe $i + 1$.

Baseado na forma como é construída a rede de fluxos N_I para um intervalo I podemos enunciar o seguinte teorema.

Teorema 4.1.3 *Os cortes na rede de fluxo N_I são um a um correspondentes a uma reclassificação local f'.*

O teorema 4.1.3 segue diretamente da construção da rede de fluxos N_I e da interpretação das arestas (u_k, u_{k+1}), pertencentes ao corte mínimo, como a associação do objeto u à classe k.

Lema 4.1.4 *Seja f' a classificação resultante de um corte mínimo C na rede de fluxos N_I. Para qualquer aresta $e = \{u, v\}$ do grafo original, a distância, resultante do corte C, entre as classes associadas a u e v é no máximo $2M$, ou seja,*

$$d(f'(u), f'(v)) \leq d((f'(u), i+1) + d(i+1, f'(v)) \leq 2M.$$

Prova.

Na rede de fluxos N_I a distância entre duas classes que pertencem ao mesmo intervalo é dada pelo número de arestas (u_k, v_k) entre elas. Por exemplo, se $f'(u) = k$ e $f'(v) = h$ e $k, h \in I$ então sua distância é dada por $|k-h|$ e, devido à forma como a rede é construída, este valor é no máximo M. Se $f'(u) \notin I$ e $f'(v) \in I$ então, $d(f'(u), f'(v)) = d((f'(u), i+1) + d(i+1, f'(v))$, onde $d((f'(u), i+1)$ é dada pela aresta (s, u_{i+1}) e é limitada por M. A quantidade $d(i+1, f'(v))$ é dada por $|i+1-f'(v)|$, que também é limitada por M. Assim a distância máxima entre $f'(u)$ e $f'(v)$ é $2M$.

□

Teorema 4.1.5 *Seja f' uma classificação resultante de um corte mínimo C em N_I. O custo da classificação f' não é maior que o custo do corte C.*

O teorema 4.1.5 segue diretamente do lema 4.1.4.

A cada iteração, a rede de fluxo é reconstruída com base nos custos obtidos na iteração anterior e é encontrado um novo corte, que representa uma nova classificação. Esta nova classificação será assumida se seu custo for menor que o custo da classificação anterior. A cada iteração é esperada uma diminuição do custo. O número de vezes em que o laço principal do algoritmo é executado é obtido analisando a melhora no custo de classificação obtida a cada iteração. Esta melhora implica no fato que quando o algoritmo *Alg_TML_GT* atingir um mínimo local, o custo esperado da classificação é de no máximo 4 vezes o custo ótimo.

Para analisar o fator de aproximação deste algoritmo, vamos fixar f^* como uma classificação ótima, e f como a classificação atual no algoritmo *Alg_TML_GT*. Para qualquer subconjunto $X \subseteq P$, seja $A^*(X)$ o custo de associação ótimo e $A(X)$ o custo de associação pago pela classificação atual no algoritmo, pago pelos objetos em X. Para o conjunto de arestas $Y \subseteq E$, $S^*(Y)$ e $S(Y)$ são os custos de separação ótimo e da classificação atual do algoritmo respectivamente. Claramente o custo da classificação ótima é dado por $Q(f^*) = A^*(P) + S^*(E)$ e o custo da classificação resultante do algoritmo é dado por $Q(f) = A(P) + S(E)$.

Considere o caso em que o algoritmo escolhe o intervalo I. Dada uma classificação ótima f^*, definimos algumas notações relacionadas ao intervalo I e à classificação f^*, que serão utilizadas na análise do algoritmo.

- P_I é o conjunto dos objetos que são associados às classes pertencentes ao intervalo I através da função de classificação f^*, ou seja, $P_I = \{u : f^*(u) \in I, \ \forall u \in P\}$.

- E_I é o conjunto das arestas $e = \{u, v\} \in E$ cujos objetos são associados à classes pertencentes ao intervalo I através da função de classificação f^*, ou seja, $E_I = \{e = \{u, v\} : f^*(u) \in I,\ f^*(v) \in I,\ \forall e \in E\}$.

- δ_I^- é o conjunto das arestas $e = \{u, v\} \in E$, cuja maior classe associada aos objetos da aresta $e = \{u, v\}$, através da função de classificação f^*, pertence ao intervalo I e a menor não, ou seja $\delta_I^- = \{e = \{u, v\} : \max(f^*(u), f^*(v)) \in I, \min(f^*(u), f^*(v)) \notin I,\ \forall e \in E\}$.

- δ_I^+ é o conjunto das arestas $e = \{u, v\} \in E$, cuja menor classe associada aos objetos da aresta $e = \{u, v\}$, através da função de classificação f^*, pertence ao intervalo I e a maior não, ou seja $\delta_I^+ = \{e = \{u, v\} : \min(f^*(u), f^*(v)) \in I, \max(f^*(u), f^*(v)) \notin I,\ \forall e \in E\}$.

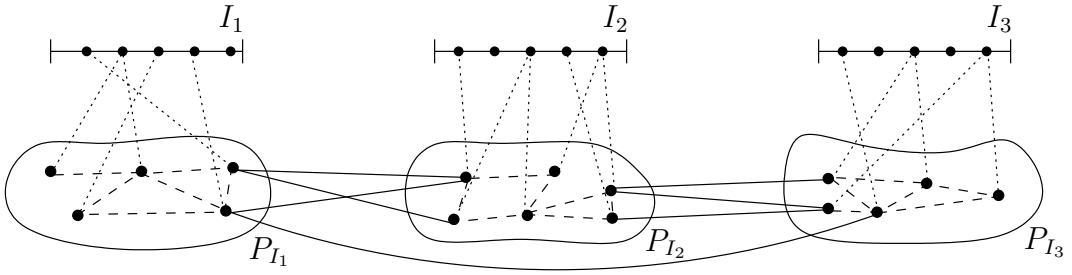

Figura 4.8: Exemplo de uma classificação f^* relacionada a um conjunto de intervalos S_r.

Para melhor entender as definições anteriores, veja a figura 4.8 que representa uma classificação f^*. Representamos por linhas pontilhadas a classificação f^* para os objetos. Como podemos observar, a união de todos os conjuntos P_{I_i} é igual ao conjunto P. As arestas tracejadas são aquelas arestas cujos objetos são associados a classes que pertencem ao mesmo intervalo, ou seja, são as arestas E_I, respectivas a cada intervalo I. As arestas entre os conjuntos P_I são as arestas de cruzamento. Note que ao considerarmos o intervalo I_2, por exemplo, as arestas que ligam objetos de P_{I_2} a objetos em P_{I_1} são as arestas que pertencem ao conjunto $\delta_{I_2}^-$ e estas mesmas arestas fazem parte do conjunto $\delta_{I_1}^+$. Assim fica claro que a união das arestas dos conjuntos δ_I^- é igual à união das arestas dos conjuntos δ_I^+, ou seja, $\cup_{I \in S_r} \delta_I^- = \cup_{I \in S_r} \delta_I^+$. Esta união das arestas de "fronteira" das partições é denotada por δ_r. Note também que $E = \delta_r \cup (\cup_{I \in S_r} E_I)$.

Lema 4.1.6 *Para uma partição S_r escolhida aleatoriamente, o valor esperado do custo de separação para as arestas de fronteira,*

$$E\left[M \sum_{e=\{u,v\} \in \delta_r} w_e\right], \text{ é } S^*(E).$$

Prova.

Seja $e = \{u, v\}$ uma aresta do grafo original, e ε_e o evento em que as classes associadas a u e v, através da função f^*, pertençam a intervalos diferentes em S_r, ou seja, ε_e é o evento em que a aresta e pertence a δ_r. A esperança de $M \sum_{e=\{u,v\}\in\delta_r} w_e$ é igual a $M \sum_{e=\{u,v\}\in E} w_e Pr[\varepsilon_e]$. A probabilidade de uma aresta $e = \{u, v\}$ fazer parte de δ_r é exatamente $d(f^*(u), f^*(v))/M$. Assim

$$\begin{aligned} E\left[M \sum_{e=\{u,v\}\in\delta_r} w_e\right] &= M \sum_{e=\{u,v\}\in E} w_e \Pr[\varepsilon_e] \\ &= M \sum_{e=\{u,v\}\in E} w_e d(f^*(u), f^*(v))/M \\ &= \sum_{e=\{u,v\}\in E} w_e d(f^*(u), f^*(v)) \\ &= S^*(E). \end{aligned}$$

□

Considere a situação em que o algoritmo escolhe um intervalo I e seja P_I o conjunto de objetos tal que $u \in P_i$ se $f(u) \in I$. Um movimento local de reclassificação possível é trocar a classificação $f(u)$, para todo $u \in P_I$, pela classificação ótima f^*, ou seja, $f(u) = f^*(u)$, para todo $u \in P_I$. Vamos utilizar este movimento local de reclassificação para limitar a melhora obtida pelo algoritmo a cada iteração.

Lema 4.1.7 *Para uma classificação f e um intervalo I, o movimento local de reclassificação, que corresponde a um corte mínimo em N_I, decrementa o custo da solução de no mínimo*

$$\left(A(P_I) + S(E_I \cup \delta_I^- \cup \delta_I^+)\right) - \left(A^*(P_I) + S^*(E_I \cup \delta_I^-) + M \sum_{e=\{u,v\}\in\delta_I^-} w_e + 2M \sum_{e=\{u,v\}\in\delta_I^+} w_e\right).$$

Prova.

Após o algoritmo ter executado o corte mínimo na rede de fluxos N_I, a classificação correspondente a este corte tem custo no máximo o valor do corte.

Para provar que a melhora não é superior a anunciada é preciso mostrar um corte com capacidade menor que o corte mínimo em N_I. Para tanto, considere um corte que gere o movimento local de reclassificação para todos os nós em P_I da sua classificação atual para a classificação f^*. Esse movimento fará com que o valor $A(P_I)$, que é o custo de associação dos objetos em P_I, passe a ser $A^*(P_I)$ e o valor $S(E_I)$, que é o custo de separação para as arestas em E_I, passe a ser $S^*(E_I)$. Para aqueles vértices e arestas que não têm sua classificação alterada neste movimento os custos de classificação e separação se mantém os mesmos.

Considere as arestas em $\delta_I^- \cup \delta_I^+$. Estas são as arestas que possivelmente terão a classificação alterada para exatamente um dos objetos a quem a aresta esta ligada. Para estimar a capacidade do corte que gerou a nova classificação f^* para os objetos em P_I, correspondente ao custo de separação destas arestas, vamos considerar as arestas em δ_I^- e δ_I^+ separadamente.

Usando o lema 4.1.4, o limitante do custo de separação para as arestas $e = \{u, v\} \in \delta_I^+$ é de $2Mw_e$. Agora, considerando uma aresta $e = \{u, v\} \in \delta_I^-$, suponha que $u \in P_I$. O limitante da capacidade das arestas em N_I pertencentes ao corte C correspondente ao custo de separação da aresta $e = \{u, v\}$, pelo lema 4.1.4, é $w_e[d(f^*(u), i+1) + d(i+1, f(v))]$. Note que devido à aresta e estar em δ_I^- temos que $f^*(v) \leq i$ e assim $d(f^*(u), i+1) \leq d(f^*(u), f^*(v))$. Limitando o outro termo $d(i+1, f(v))$ por M obtemos que a capacidade correspondente à separação dos objetos ligados à aresta $e = \{u, v\}$ é no máximo $w_e[d(f^*(u), f^*(v)) + M]$. Assim temos

$$\begin{aligned} & A^*(P_I) + S^*(E_I) + \sum_{e=\{u,v\}\in\delta_I^-} w_e(d(f^*(u), f^*(v)) + M) + \sum_{e\in\delta_I^+} 2Mw_e \\ = \; & A^*(P_I) + S^*(E_I) + \sum_{e=\{u,v\}\in\delta_I^-} w_e d(f^*(u), f^*(v)) + \sum_{e=\{u,v\}\in\delta_I^-} w_e M + \sum_{e\in\delta_I^+} 2Mw_e \\ = \; & A^*(P_I) + S^*(E_I \cup \delta_I^-) + M \sum_{e\in\delta_I^-} w_e + 2M \sum_{e\in\delta_I^+} w_e. \end{aligned}$$

□

Teorema 4.1.8 *Se o algoritmo alcançou uma classificação f e esta classificação representa um mínimo local, então o custo $Q(f)$ é no máximo 4 vezes o custo ótimo $Q(f^*)$.*

Prova.

O fato do algoritmo ter alcançado uma classificação f que é um mínimo local implica que a garantia de melhora do lema 4.1.7 neste momento é não positiva para qualquer intervalo I, ou seja,

$$A(P_I) + S(E_I \cup \delta_I^- \cup \delta_I^+) \leq A^*(P_I) + S^*(E_I \cup \delta_I^-) + M \sum_{e=\{u,v\}\in\delta_I^-} w_e + 2M \sum_{e=\{u,v\}\in\delta_I^+} w_e.$$

Agora, considerando a partição S_r e somando estas desigualdades para cada $I \in S_r$ temos do lado esquerdo $A(P) + S(E) + S(\delta_r)$, uma vez que as arestas em δ_r ocorrem nas fronteiras de dois intervalos. Esta quantia é pelo menos $Q(f)$, o custo da classificação f. Somando o lado direito, temos exatamente $A^*(P) + S^*(E) + 3M \sum_{e=\{u,v\}\in\delta_r} w_e$. Por conseqüência temos

$$Q(f) \leq A^*(P) + S^*(E) + 3M \sum_{e=\{u,v\}\in\delta_r} w_e,$$

para qualquer partição S_r. Baseado nisso, pode-se obter a esperança do custo de classificação. O lado esquerdo da desigualdade é constante, e pelo lema 4.1.6, o custo esperado do lado direito

é no máximo $A^*(P) + 4S^*(E)$. Assim temos $Q(f) \leq A^*(P) + 4S^*(E) \leq 4Q(f^*)$ conforme anunciado.

□

Teorema 4.1.9 *Seja Q_0 o custo da classificação inicial. Se o laço principal do algoritmo Alg_TML_GT for repetido $O((m/M)(\log Q_0 + \log \epsilon^-))$ vezes, então o custo esperado da classificação resultante é no máximo $(4 + \epsilon)Q(f^*)$.*

Prova.

Para estimar o decréscimo esperado do custo de classificação em uma iteração, considere a seguinte alternativa para seleção aleatória de um intervalo. Primeiro, selecione aleatoriamente uma partição S_r e em seguida selecione um intervalo $I \in S_r$. Este processo seleciona cada intervalo, para ser usado pelo algoritmo *Alg_TML_GT*, com aproximadamente a mesma probabilidade, a diferença é que algumas partições têm um intervalo a mais que outras, e a probabilidade de escolher um intervalo nestas partições é menor.

Considerando uma classificação f, uma partição S_r e a escolha do intervalo I, o lema 4.1.7 é usando para estimar um limitante da melhoria obtida para o intervalo I. Como na prova anterior, é somada a estimativa de melhora para todos os intervalos da partição S_r.

Como há no máximo $\lceil m/M \rceil + 1$ intervalos em S_r, o decréscimo esperado no custo da classificação, quando um intervalo é selecionado aleatoriamente de S_r, é pelo menos

$$\frac{1}{\lceil m/M \rceil + 1} \left(Q(f) - Q(f^*) - 3M \sum_{e=\{u,v\} \in \delta_r} w_e \right).$$

Mas pelo lema 4.1.6, para uma partição S_r escolhida aleatoriamente, o valor esperado desse decréscimo é pelo menos

$$\Omega \left(\frac{M}{m} (Q(f) - 4Q(f^*)) \right),$$

assim, em $O(m/M)$ iterações é esperada que a diferença $Q(f) - 4Q(f^*)$ decresça por um fator constante. Isto implica no limite do tempo de execução anunciado.

□

Capítulo 5

Algoritmo Guloso para o Problema de Classificação Métrica Uniforme

Neste capítulo apresentamos um novo algoritmo, que é experimentalmente polinomial, para o problema de classificação métrica uniforme UML. A idéia utilizada na criação deste algoritmo surgiu do estudo de problemas correlatos, como o problema de localização de recursos e o problemas da cobertura por conjuntos. Antes de mostrar o novo algoritmo, vamos apresentar uma prova de um fator de aproximação de um algoritmo guloso, para o problema de cobertura por conjuntos, via uma análise primal-dual, em seguida mostraremos uma abordagem para o problema de localização de recursos, que serviu de inspiração para o desenvolvimento deste novo algoritmo. Também, mostraremos uma família de instâncias cujo o fator de aproximação é $\Omega(\log n)$.

5.1 Uma Análise Primal-Dual para o Problema de Cobertura por Conjuntos

Nesta seção apresentamos uma versão primal-dual para um algoritmo aproximado guloso para o problema de cobertura por conjuntos, incluindo uma prova de seu fator de aproximação. Esta versão primal-dual difere da maioria dos algoritmos primal-duais por usar uma atribuição α, que pode ser dual inviável, e no final busca um valor γ, tal que, α/γ seja uma solução dual viável. Apresentamos, também, nesta seção uma generalização da prova para o algoritmo primal-dual para o caso em que os custos dos conjuntos sejam aproximados.

Uma instância para o problema de Cobertura por Conjuntos Mínima (MinCC) é composta de um conjunto $E_{\text{SC}} = \{e_1, e_2, \ldots, e_n\}$, de n objetos, uma coleção $\mathcal{S}_{\text{SC}} = \{C_1, \ldots, C_t\}$, de subconjuntos de E_{SC}, e uma função de custo $w : \mathcal{S}_{\text{SC}} \rightarrow \mathbb{Q}^+$. O objetivo é encontrar uma coleção $\mathcal{S}_{\text{SC}}{}' \subseteq \mathcal{S}_{\text{SC}}$, tal que $\cup_{C_i \in \mathcal{S}_{\text{SC}}{}'} C_i = E_{\text{SC}}$, que minimize $\sum_{C_i \in \mathcal{S}_{\text{SC}}{}'} w(C_i)$.

Um algoritmo guloso clássico para o problema de cobertura por conjuntos é apresentado por Chvátal [8], e é uma H_g-aproximação, onde g é o número de elementos do maior conjunto em $\mathcal{S}_{\text{SC}}$ e H_g é o valor do número harmônico de grau g. Neste algoritmo temos um conjunto U dos elementos "não cobertos", que são aqueles que não pertencem a qualquer conjunto que já esteja na solução. Inicialmente o conjunto U contém todos os elementos. A cada iteração o algoritmo escolhe um conjunto $C_j \in \mathcal{S}_{\text{SC}}$ que tenha o menor custo amortizado, que é o custo w_j do conjunto C_j dividido pelo número de elementos "não cobertos", ou seja, $\frac{w_j}{C_j \cap U}$. Uma vez que um conjunto é escolhido para fazer parte da solução, todos os elementos deste conjunto são considerados "cobertos" e então removidos do conjunto U. Na figura 5.1 descrevemos este algoritmo mais formalmente.

ALGORITMO *Alg_GCC* $(E_{\text{SC}}, \mathcal{S}_{\text{SC}}, w)$

onde E_{SC} é um conjunto de elementos, $\mathcal{S}_{\text{SC}}$ é uma família de conjuntos e

w é o custo dos conjuntos.

11. $Sol \leftarrow \emptyset$; $U \leftarrow E_{\text{SC}}$.
12. Enquanto $U \neq \emptyset$ faça
13. $j' \leftarrow \arg\min_{j: C_j \cap U \neq \emptyset} \frac{w_j}{|C_j \cap U|}$
14. $Sol \leftarrow Sol \cup \{j'\}$
15. $U \leftarrow U \setminus C_{j'}$
16. Devolva Sol.

Figura 5.1: Algoritmo guloso para o problema de cobertura por conjuntos.

Para mostrar o algoritmo primal-dual para o problema de cobertura por conjuntos, apresentaremos primeiramente uma formulação que usa variáveis binárias x_j, para cada conjunto $C_j \in \mathcal{S}_{\text{SC}}$, onde $x_j = 1$, se e somente se, C_j é escolhido para entrar na solução. A formulação consiste em encontrar x que

$$\begin{array}{lll} \min & \sum_j w_j x_j & \\ s.a & \sum_{j: e \in C_j} x_j \geq 1 & \forall\ e \in E_{\text{SC}}, \\ & x_j \in \{0,1\} & \forall\ j \in \{1,\ldots,t\}, \end{array}$$

e o dual da versão relaxada consiste em encontrar α que

$$\begin{array}{lll} \min & \sum_{e \in E_{\text{SC}}} \alpha_e & \\ s.a & \sum_{e \in C_j} \alpha_e \leq w_j & \forall\ j \in \{1,\ldots,t\}, \\ & \alpha_e \geq 0 & \forall\ e \in E_{\text{SC}}. \end{array}$$

O algoritmo guloso pode ser reescrito como um algoritmo primal-dual com um conjunto similar de eventos onde os elementos são inseridos na solução na mesma ordem. Para reescrevermos o algoritmo usaremos um conjunto U, que contém os elementos não cobertos, inicialmente igual ao conjunto E_{SC}. A cada iteração, o conjunto U é atualizado removendo-se os elementos "cobertos" na iteração. Além do conjunto U, usaremos uma variável T, para expressar a noção de tempo associado a cada evento, e usaremos variáveis duais α_e, associadas a cada elemento $e \in E_{\mathrm{SC}}$. As variáveis α_e têm seus valores iniciais iguais a zero e são aumentadas uniformemente a cada iteração, juntamente com a variável T. O algoritmo primal-dual está ilustrado na figura 5.2.

ALGORITMO *Alg_PDCC* $E_{\mathrm{SC}}, \mathcal{S}_{\mathrm{SC}}, w)$

onde E_{SC} é um conjunto de elementos, $\mathcal{S}_{\mathrm{SC}}$ é uma família de conjuntos e

w o custo dos conjuntos.

1. $U \leftarrow E_{\mathrm{SC}}$; $T \leftarrow 0$; $\alpha_e \leftarrow 0, \forall e \in E_{\mathrm{SC}}$; $Sol \leftarrow \emptyset$.

2. Enquanto $U \neq \emptyset$ faça

3. Aumente uniformemente o tempo T e as variáveis $\alpha_e : e \in U$ até existir um índice j tal que $\sum_{e \in C_j \cap U} \alpha_e$ seja igual a w_j.

4. $Sol \leftarrow Sol \cup \{j\}$.

5. $U \leftarrow U \setminus C_j$.

6. Devolva Sol.

Figura 5.2: Algoritmo primal-dual para o problema de cobertura por conjuntos.

Lema 5.1.1 *A seqüência de eventos executada pelo algoritmo guloso Alg_GCC e pelo algoritmo Primal-Dual Alg_PDCC é a mesma.*

Prova.

Note que o valor de α_e quando $\sum_{e \in C_j \cap U} \alpha_e = w_j$ é igual ao custo amortizado do conjunto C_j. Desde que α_e cresça uniformemente, no passo 3 do algoritmo *Alg_PDCC*, o algoritmo escolhe o conjunto com menor custo amortizado em cada iteração.

□

Este lema indica que todas as soluções obtidas pelo algoritmo guloso podem ser analisadas através da técnica primal-dual associada ao algoritmo *Alg_PDCC*.

Lema 5.1.2 *Seja $C_j = \{e_1, e_2, \ldots, e_k\}$ e α_i a variável de tempo associada ao item e_i, obtida no final do algoritmo primal-dual. Se $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k$, então em um instante α_l, para todo $1 \leq l \leq k$, $\sum_{i=l}^{k} \alpha_l \leq w_j$.*

Prova.

No momento exatamente anterior ao tempo α_l, todas as variáveis α_i, $l \leq i \leq k$ tem o mesmo valor, α_l, e todas elas são associadas a um elemento não coberto. Suponha que o lema é falso. Neste caso, $\sum_{i=l}^{k} \alpha_l > w_j$ e em um instante imediatamente anterior a α_l, o conjunto C_j deveria ter entrado na solução e todos os seus elementos deveriam ter $\alpha < \alpha_l$, o que é uma contradição, já que C_j tem pelo menos um elemento maior ou igual a α_l. Assim, o lema é válido.

□

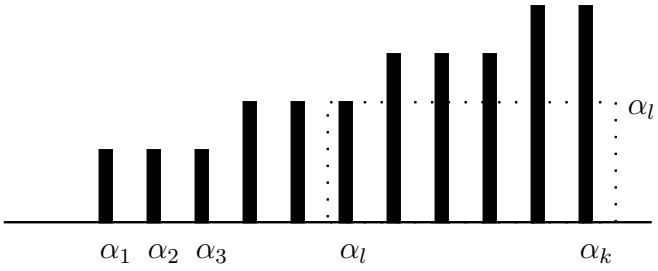

Figura 5.3: Exemplo dos valores de α no momento α_l.

O algoritmo primal-dual devolve uma solução primal Sol, de custo $val(Sol) = \sum_{e \in E} \alpha_e$, que é viável e inteira. Note que uma solução α pode ser dual inviável, devido ao passo 3 do algoritmo *Alg_PDCC* aumentar apenas as variáveis α_e cujo elemento e não tenha sido coberto ainda até que $\sum_{e \in C_j \cap U} \alpha_e = w_j$, fazendo com que a restrição $\sum_{e \in C_j} \alpha_e \leq w_j$ possa ser violada. Se existir um valor γ tal que α/γ é dual viável, ou seja, $\sum_{e \in C_j} \alpha_e/\gamma \leq w_j$, para cada $j \in \{1, \ldots, t\}$, então, pelo teorema forte da dualidade, $val(Sol) \leq \gamma OPT$. A idéia usada para provar o fator de aproximação é encontrar um valor γ tal que α/γ seja dual viável.

Teorema 5.1.3 *O algoritmo primal-dual para o problema de cobertura por conjuntos é um algoritmo H_g-aproximado, onde g é o tamanho do maior conjunto em $\mathcal{S}_{\mathrm{SC}}$ e H_g é o valor do número harmônico de grau g.*

Prova.

Considere um conjunto arbitrário $C = \{e_1, \ldots, e_k\}$, com k elementos e com custo w. Cada elemento $e_i \in C$ possui uma variável de tempo α_i associada, para $i = 1, \ldots, k$. Sem perda de generalidade , assuma que $\alpha_1 \leq \ldots \leq \alpha_k$.

Se γ é um valor tal que α/γ é dual viável então

$$\sum_{i=1}^{k} \alpha_i/\gamma \leq w, \tag{5.1}$$

assim, uma condição necessária é

$$\gamma \geq \frac{\sum_{i=1}^{k} \alpha_i}{w}. \tag{5.2}$$

Aplicando o lema 5.1.2 para cada valor de l, temos

$$\begin{cases} l=1, & k\alpha_1 \leq w, & \Rightarrow & \alpha_1/w \leq 1/k, \\ l=2, & (k-1)\,\alpha_2 \leq w, & \Rightarrow & \alpha_2/w \leq 1/(k-1), \\ & & \vdots & \\ l=k, & \alpha_k \leq w, & \Rightarrow & \alpha_k/w \leq 1. \end{cases}$$

Somando as desigualdades acima temos

$$\frac{\sum_{i=1}^{k} \alpha_i}{w} \leq H_k. \tag{5.3}$$

Conseqüentemente, quando $\gamma = H_g$ temos que α/γ é uma solução dual viável.

□

Uma instância que mostra que este fator de aproximação é justo está ilustrada na figura 5.4. Nesta instância temos n elementos $E_{\text{SC}} = \{e_1, e_2, \ldots, e_n\}$, e $n+1$ conjuntos que pertencem a família de conjuntos $\mathcal{S}_{\text{SC}} = \{C_1, \ldots, C_{n+1}\}$, onde a composição dos conjuntos é a seguinte: $C_1 = \{e_1\}$, $C_2 = \{e_2\}$, $\ldots$, $C_n = \{e_n\}$ e $C_{n+1} = \{e_1, e_2, \ldots, e_n\}$ com os custos $\frac{1}{n}$, $\frac{1}{n-1}$, $\ldots$, $\frac{1}{2}$, 1, e $1+\epsilon$ respectivamente.

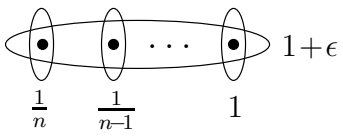

Figura 5.4: Instância justa para o algoritmo *Alg_PDCC*.

Quando executamos o algoritmo *Alg_PDCC* sobre esta instância a solução devolvida é composta dos n conjuntos unitários. Assim o custo da solução devolvida pelo algoritmo é

$$\frac{1}{n} + \frac{1}{n-1} + \ldots + 1 = H_n,$$

enquanto que o custo da cobertura ótima é $1+\epsilon$.

Agora, considere uma pequena modificação no algoritmo *Alg_GCC*. No passo 3 do algoritmo guloso, ao invés de escolher o conjunto com menor custo amortizado, passamos a escolher um conjunto C_j com custo amortizado no máximo f vezes maior que o custo de um conjunto de menor custo amortizado. Se C_{j*} é um conjunto com o menor custo amortizado então a seguinte desigualdade é válida.

$$\frac{w_j}{|C_j \cap U|} \leq f \frac{w_{j*}}{|C_{j*} \cap U|}.$$

Esta modificação permite que o custo amortizado do conjunto escolhido seja no máximo f vezes maior que o menor custo amortizado de um conjunto. Denotamos por $\mathcal{A}_f$ o algoritmo primal-dual com esta modificação.

Lema 5.1.4 *Seja $C_j = \{e_1, e_2, \ldots, e_k\}$ e α_i a variável de tempo associada ao item e_i gerada pelo algoritmo $\mathcal{A}_f$. Se $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k$ então em um instante α_l, para todo $1 \leq l \leq k$, $\sum_{i=l}^{k} \alpha_l \leq f\, w_j$.*

Prova.

Suponha por contradição que o lema é falso. Neste caso, existe uma execução onde $\sum_{i=l}^{k} \alpha_l > f\, w_j$ e, em um instante $T < \alpha_l$, o conjunto C_j teria sido escolhido para entrar na solução impedindo que os valores de α_i chegassem ao valor α_l ou maior, o que é uma contradição.

□

Teorema 5.1.5 *Se g é o tamanho do maior conjunto em $\mathcal{S}_{\text{SC}}$, então o algoritmo $\mathcal{A}_f$ é um algoritmo $f\, H_g$-aproximado.*

Prova.

Considere um conjunto arbitrário $C = \{e_1, \ldots, e_k\}$, com k elementos e com custo w. Cada elemento $e_i \in C$ possui uma variável de tempo α_i associada, para $i = 1, \ldots, k$. Pela descrição do algoritmo $\mathcal{A}_f$, a soma $\sum_{e \in C} \alpha_e$ pode ultrapassar o valor de w por no máximo um fator de f. Sem perda de generalidade, assuma que $\alpha_1 \leq \ldots \leq \alpha_k$.

Se γ é um valor tal que α/γ é dual viável então

$$\sum_{i=1}^{k} \alpha_i/\gamma \leq w, \tag{5.4}$$

assim, uma condição necessária é

$$\gamma \geq \frac{\sum_{i=1}^{k} \alpha_i}{w}. \tag{5.5}$$

Aplicando o lema 5.1.4 para cada valor de l, temos

$$\begin{cases} l = 1, & k\alpha_1 \leq f \cdot w, & \Rightarrow & \alpha_1/w \leq f \cdot (1/k), \\ l = 2, & (k-1)\,\alpha_2 \leq f \cdot w, & \Rightarrow & \alpha_2/w \leq f \cdot (1/(k-1)), \\ & & \vdots & \\ l = k, & \alpha_k \leq f \cdot w, & \Rightarrow & \alpha_k/w \leq f \cdot (1). \end{cases}$$

Somando as desigualdades acima temos

$$\frac{\sum_{i=1}^{k} \alpha_i}{w} \leq f \cdot H_k. \tag{5.6}$$

Conseqüentemente, quando $\gamma = f \cdot H_g$ temos que α/γ é uma solução dual viável. Assim, pelo teorema forte da dualidade, o algoritmo $\mathcal{A}_f$ é um algoritmo $f\, H_g$-aproximado.

□

5.2 O Problema de Localização de Recursos

A idéia que inspirou um novo algoritmo para o problema de classificação métrica uniforme surgiu ao estudarmos um algoritmo apresentado por Jain *et al.* [14], para o problema de localização de recursos, que utiliza a idéia de estrelas. Uma *estrela* é um grafo conexo onde apenas um vértice, chamado centro da estrela, pode ter grau maior que um e um conjunto de estrelas é denominado *constelação*. A figura 5.2 ilustra um conjunto de estrelas.

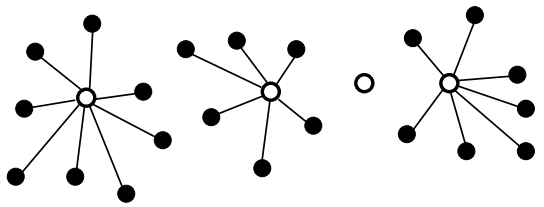

Figura 5.5: Conjunto de estrelas.

Uma estrela, para o problema de localização de recursos, é composta por um recurso, que é o nó central, e os clientes atendidos por este recurso. Já uma estrela para o problema de classificação métrica é formada por uma classe, que é o nó central, e os objetos classificados como pertencentes a esta classe.

A idéia de estrelas no problema de localização de recursos surgiu com um algoritmo de Jain e Vazirani [15], que é baseado na análise da formulação dual do problema de localização de recursos. Uma solução para o problema de localização de recursos é um par (I, ϕ), onde $I \subseteq F$ é o conjunto dos recursos abertos e $\phi(j)$ é uma função que indica o recurso em I que atende o cliente j.

Jain *et al.* [14] apresentam uma forma de resolver o problema de localização de recursos utilizando a idéia de estrelas, semelhante à usada por Jain e Vazirani [15]. O algoritmo é iterativo e trabalha de forma gulosa, escolhendo sempre a estrela de menor custo amortizado. O custo amortizado de uma estrela, para o problema de localização de recursos, é dado pela relação entre o custo da estrela e o número de clientes atendidos por ela. O custo de uma estrela é dado por dois fatores: o custo de abrir o recurso e o custo de atender os clientes usando aquele recurso. Mais formalmente, o custo de uma estrela (i, C'), onde i é o recurso e $C' \subseteq C$ é um subconjunto de clientes, é dado pelo valor $f_i + \sum_{j \in C'} c_{ij}$. O custo amortizado de uma estrela (i, C') é a razão entre o custo da estrela e o tamanho de C', ou seja, $(f_i + \sum_{j \in C'} c_{ij})/|C'|$.

Na descrição do algoritmo de Jain *et al.* [14], que chamamos por *Alg_SFL*, denotamos por U o conjunto dos clientes não atendidos por nenhum recurso, por $\mathcal{S}_{\text{FLP}}$ o conjunto de todas as estrelas para o conjunto U, por A o conjunto dos recursos abertos e por ϕ a função de conexão de um cliente a um recurso. O algoritmo de Jain *et al.* [14] é apresentado na figura 5.6.

Se observarmos os passos 4 e 5 do algoritmo *Alg_SFL*, onde são geradas todas as estrelas para o conjunto U e entre estas estrelas é escolhida uma de menor custo efetivo, podemos

ALGORITMO *Alg_SFL* (G)
onde G é um grafo bipartido (F, C).
1. $A \leftarrow \emptyset$
2. $U \leftarrow C$
3. Enquanto $U \neq \emptyset$
4. $\quad \mathcal{S}_{\text{FLP}} \leftarrow \{(i, C) : C \subseteq U,\ \ i \in F\}$
5. $\quad$ Escolha uma estrela $S = (i, C')$ de menor custo amortizado em $\mathcal{S}_{\text{FLP}}$
6. $\quad$ Se $i \notin A$
7. $\quad\quad$ Pague o custo f_i
8. $\quad\quad A \leftarrow A \cup f_i$
9. $\quad\quad f_i \leftarrow 0$
10. $\quad$ Para todo $j \in C'$ faça
11. $\quad\quad \phi(j) = i$
12. $\quad U \leftarrow U \backslash C'$
13. Devolva (A, ϕ)

Figura 5.6: Algoritmo guloso para o problema de localização de recursos usando estrelas.

concluir que em princípio o algoritmo *Alg_SFL* possui tempo de execução exponencial. Isto ocorre porque cada cliente pode ou não estar ligado a um recurso, assim temos que o número de estrelas com pelo menos um cliente é $n_f * (2^{n_c} - 1)$, onde n_f é o número de recursos e n_c é o número de clientes.

Observando uma solução para uma instância do problema, temos que apenas as estrelas de menor custo amortizado são utilizadas. Portanto, se for possível encontrar apenas as estrelas de menor custo amortizado em tempo polinomial, o algoritmo *Alg_SFL* será polinomial. Jain *et al.* [14], observaram que apenas os clientes mais próximos são atendidos pelo recurso i, ou seja, a estrela de menor custo amortizado é formada por aqueles clientes que têm menor custo de associação. A figura 5.7 mostra um exemplo de estrela formada pelos clientes mais próximos, o que resulta no menor custo amortizado.

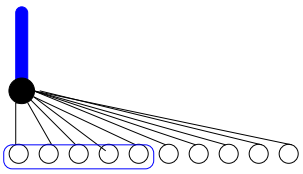

Figura 5.7: Exemplo de estrela de menor custo amortizado.

Uma forma de encontrar as estrelas de menor custo amortizado, para um recurso $i \in F$, é ordenar os custos de associação c_{ij}, para todo $j \in C$. Sem perda de generalidade, considere

$c_{i1} \leq c_{i2} \leq \ldots \leq c_{in}$, e escolha os k clientes que possuem os menores custos c_{ij}, tal que o valor da equação $(f_i + \sum_{j=1}^{k} c_{ij})/k$ seja minimizado. Como a ordenação e a escolha dos k menores elementos pode ser feita em tempo polinomial, a escolha de uma estrela de menor custo amortizado pode ser feita em tempo polinomial. Conseqüentemente há uma maneira de implementar o algoritmo *Alg_SFL* descrito de forma que este consuma tempo polinomial.

5.3 Um Algoritmo Primal-Dual para o Problema de Classificação Métrica Uniforme

O algoritmo que apresentaremos para o problema de classificação métrica uniforme, usa uma idéia similar ao algoritmo apresentado por Jain *et al.*[14] para o problema de localização de recursos, apresentado na seção anterior. No problema de classificação métrica podemos considerar uma classificação $f : P \rightarrow L$ como um conjunto de estrelas, onde cada estrela é formada por uma classe, que é nó central, e os objetos associados a esta classe. Uma estrela S para o problema de classificação métrica uniforme é uma tupla (l, U_S), onde $l \in L$ e $U_S \subseteq P$. O novo algoritmo para o problema de classificação métrica uniforme seleciona a cada iteração uma estrela de menor custo amortizado, até que todos os objetos tenham sido classificados.

O custo $\mathcal{C}(S)$ de uma estrela $S = (l, U_S)$, onde $l \in L$ e $U_S \subseteq P$, é definido como

$$\mathcal{C}(S) = c(U_S) + \frac{1}{2} w(U_S),$$

onde $c(U_S) = \sum_{u \in U_S} c_{ul}$ e $w(U_S) = \sum_{u \in U_S, v \in (P \setminus U_S)} w_{uv}$, que é o custo de associar cada elemento de U_S à classe l somado com metade do custo de separar cada objeto de U_S dos objetos em $P \setminus U_S$. É considerada apenas a metade do custo de separação porque a outra metade será considerada quando os objetos em $P \setminus U_S$ forem associados a alguma estrela.

A figura 5.8 ilustra uma instância para o problema de classificação métrica uniforme e a figura 5.9 ilustra todas as possíveis estrelas e seus custos para esta instância.

Os principais passos do algoritmo consistem em:

a) Gerar estrelas de menor custo amortizado para todas as classes;

b) Escolher entre todas as estrelas geradas uma estrela de menor custo relativo;

c) Associar os objetos pertencentes a esta estrela à respectiva classe;

d) Remover os objetos associados na iteração do conjunto dos objetos considerados;

e) Repetir os passos acima até que todos os objetos tenham sido classificados.

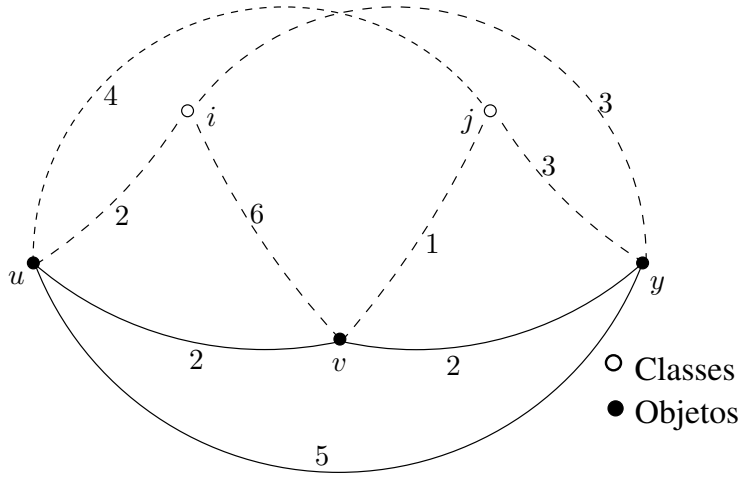

Figura 5.8: Uma instância para o problema UML.

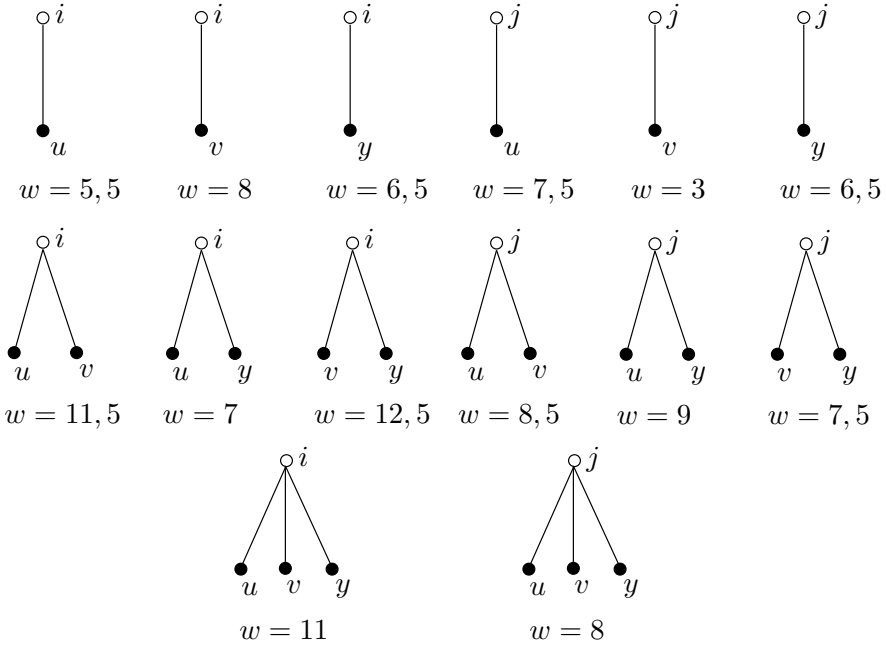

Figura 5.9: Todas as estrelas da instância apresentada na figura 5.8.

A busca por estrelas de menor custo amortizado é resolvida através de um algoritmo para o problema de cobertura por conjuntos. Para isso é preciso mapear as estrelas, geradas para o problema de classificação métrica uniforme, em uma instância para o problema de cobertura por conjuntos.

A transformação do conjunto de todas as possíveis estrelas $\mathcal{S}$, de uma instância do problema de classificação métrica uniforme, em uma instância $(E_{\mathrm{SC}}, \mathcal{S}_{\mathrm{SC}}, w')$ do problema de cobertura

por conjuntos pode ser feita da seguinte forma:

- O conjunto E_{SC}, dos elementos a serem cobertos, é igual ao conjunto P de objetos.
- A família de conjuntos $\mathcal{S}_{\text{SC}} = \{C_1, \ldots, C_t\}$ é formada pelas estrelas S, onde $S = (l, U_S)$, para todo $l \in L$ e $U_S \subseteq P$.
- O custo w'_i de cada conjunto $C_i \in \mathcal{S}_{\text{SC}}$ é o custo da estrela $S = (l, U_S)$ que originou o conjunto C_i.

A instância para o problema UML, apresentada na figura 5.8, e as estrelas para esta instância, apresentadas na figura 5.9, geram uma instância para o problema de cobertura por conjuntos, onde o conjunto de elementos E_{SC} é igual a $\{u, v, y\}$ e a coleção de conjuntos $\mathcal{S}_{\text{SC}}$ pode ser vista na figura 5.10.

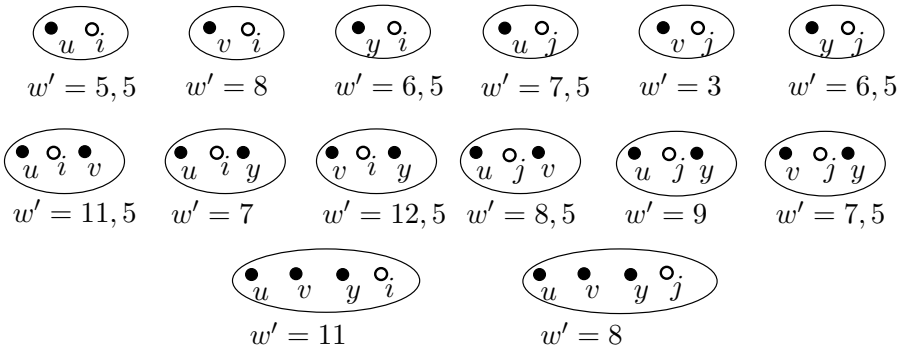

Figura 5.10: Todos os conjuntos gerados pelas estrelas apresentadas na figura 5.9.

Na figura 5.11 é apresentada a descrição do algoritmo guloso para o problema de classificação uniforme, GUL, que usa um algoritmo aproximado para o problema de cobertura por conjuntos. Assumimos que um dos parâmetros de entrada do algoritmo GUL é o algoritmo aproximado para o problema de cobertura por conjuntos que será utilizado no algoritmo. Na descrição do algoritmo GUL usamos a seguinte notação:

- $\mathcal{S}_{\text{ULP}}$ é o conjunto de todas as possíveis estrelas;
- U é o conjunto dos objetos não classificados;
- $(E_{\text{SC}}, \mathcal{S}_{\text{SC}}, w')$ é uma instância para o problema de cobertura por conjuntos;
- $\mathcal{U}$ é uma coleção de conjuntos de objetos;
- f é uma classificação.

ALGORITMO GREEDY UNIFORM LABELING (GUL) $(P, L, d, c, w, \mathcal{A}_{\text{SC}})$

onde (P, L, d, c, w) é uma instância para o problema UML e

$\mathcal{A}_{\text{SC}}$ é um algoritmo β-aproximado para o problema de cobertura por conjuntos.

1. $E_{\text{SC}} \leftarrow P$.
2. $\mathcal{S}_{\text{SC}} \leftarrow \{U_S : S = (l, U_S) \in \mathcal{S}_{\text{ULP}}\}$.
3. $w'_S \leftarrow \mathcal{C}_S + \frac{1}{2}w(U_S), \quad \forall\ S = (l, U_S) \in \mathcal{S}_{\text{ULP}}$.
4. $\mathcal{U} \leftarrow \mathcal{A}_{\text{SC}}\,(E_{\text{SC}}, \mathcal{S}_{\text{SC}}, w')$, seja $\mathcal{U} = \{U_{S_1}, U_{S_2}, \ldots, U_{S_t}\}$.
5. $U \leftarrow P$.
6. Para $k \leftarrow 1$ até t faça
7. Seja $l \in L$ a classe associada com U_{S_k};
8. $f(i) \leftarrow l, \forall i \in U_{S_k} \cap U$;
9. $U \leftarrow U \setminus U_{S_k}$.
10. devolva f.

Figura 5.11: Novo algoritmo para o problema de classificação métrica uniforme.

5.3.1 Análise do Algoritmo

Para analisar o algoritmo usaremos a seguinte notação:

- $val_{\text{ULP}}(f)$ é o valor, no problema de classificação métrica uniforme, da classificação f;
- $val_{\text{SC}}(\mathcal{U})$ é o valor, no problema de cobertura por conjuntos, da coleção $\mathcal{U}$;
- f_{OPT} é uma classificação ótima para o problema de classificação métrica uniforme;
- $\mathcal{U}_{\text{OPT}}$ é uma solução ótima para o problema de cobertura por conjuntos;
- $SC(f)$ é uma coleção $\{U_{S_1}, U_{S_2}, \ldots, U_{S_k}\}$ relacionada com uma classificação $f = \{S_1, S_2, \ldots, S_k\}$, onde $S_i = (l, U_{S_i})$ é uma estrela.

Lema 5.3.1 *Se f é uma solução devolvida pelo algoritmo* GUL *e $\mathcal{U}$ é a solução devolvida pelo algoritmo $\mathcal{A}_{\text{SC}}$ para o problema de cobertura por conjuntos (no passo 4) então*

$$val_{\text{ULP}}(f) \leq val_{\text{SC}}(\mathcal{U}).$$

Prova.

$$
\begin{aligned}
val_{\mathrm{ULP}}(f) &= \sum_{S\in f}\mathcal{C}(S) = \sum_{S\in f}\left(c(U_S)+\frac{1}{2}w(U_S)\right) && (5.7)\\
&\leq \sum_{U_S\in\mathcal{U}}\left(c(U_S)+w(U_S)\right) && (5.8)\\
&= \sum_{U_S\in\mathcal{U}}\left(\mathcal{C}(S)+\frac{1}{2}w(U_S)\right)\\
&= \sum_{U_S\in\mathcal{U}} w'_S = val_{\mathrm{SC}}(\mathcal{U})
\end{aligned}
$$

A igualdade (5.7) é válida, já que as estrelas em f são disjuntas, isto pode ser checado por contagem. A desigualdade (5.8) é válida, já que um custo de associação de $val_{\mathrm{ULP}}(f)$ também aparece em $val_{\mathrm{SC}}(\mathcal{U})$,

$$\sum_{S\in f} c(U_S) \leq \sum_{U_S\in\mathcal{U}} c(U_S),$$

e o custo de separação w_{uv}, para qualquer u e v, onde $f(u)\neq f(v)$, que aparece em $val_{\mathrm{ULP}}(f)$ deve aparecer, uma ou duas vezes, em $val_{\mathrm{SC}}(\mathcal{U})$,

$$\sum_{S\in f}\frac{1}{2}w(U_S) = \sum_{u<v:f(u)\neq f(v)} w_{uv} \leq \sum_{U_S\in\mathcal{U}} w(U_S).$$

□

Lema 5.3.2 $val_{\mathrm{SC}}(\mathcal{U}_{\mathrm{OPT}}) \leq 2val_{\mathrm{ULP}}(f_{\mathrm{OPT}})$.

Prova.

$$
\begin{aligned}
2val_{\mathrm{ULP}}(f_{\mathrm{OPT}}) &= 2\sum_{S\in f_{\mathrm{OPT}}}\mathcal{C}(S) \geq \sum_{S\in f_{\mathrm{OPT}}}\left(\mathcal{C}(S)+\frac{1}{2}w(U_S)\right) && (5.9)\\
&= \sum_{t\in SC(f_{\mathrm{OPT}})} w'_t\\
&\geq \sum_{t\in\mathcal{U}_{\mathrm{OPT}}} w'_t && (5.10)\\
&= val_{\mathrm{SC}}(\mathcal{U}_{\mathrm{OPT}}),
\end{aligned}
$$

onde a desigualdade (5.9) é válida, já que $\mathcal{C}(S) \geq \frac{1}{2}w(U_S)$. A a desigualdade (5.10) é válida, já que $SC(f_{\mathrm{OPT}})$ é uma solução para o problema cobertura por conjuntos, mas não necessariamente com valor ótimo. □

Teorema 5.3.3 *Se I é uma instância para o problema UML, f é a classificação gerada pelo algoritmo* GUL *e f_{OPT} é uma classificação ótima para I então*

$$val_{\mathrm{ULP}}(f) \leq 2\beta\, val_{\mathrm{ULP}}(f_{\mathrm{OPT}}),$$

onde β é o fator de aproximação do algoritmo $\mathcal{A}_{\mathrm{SC}}$ dado como parâmetro.

Prova.

Seja $\mathcal{U}$ a solução devolvida pelo algoritmo $\mathcal{A}_{\mathrm{SC}}$ (passo 4 do algoritmo GUL) e $\mathcal{U}_{\mathrm{OPT}}$ uma solução ótima para a instância do problema de cobertura por conjuntos correspondente. Neste caso temos

$$\begin{aligned} val_{\mathrm{ULP}}(f) &\leq val_{\mathrm{SC}}(\mathcal{U}) && (5.11)\\ &\leq \beta\, val_{\mathrm{SC}}(\mathcal{U}_{\mathrm{OPT}}) && (5.12)\\ &\leq 2\beta\, val_{\mathrm{ULP}}(f_{\mathrm{OPT}}), && (5.13) \end{aligned}$$

onde a desigualdade (5.11) é válida pelo lema 5.3.1, a desigualdade (5.12) é válida já que $\mathcal{U}$ é encontrada por um algoritmo β-aproximado e a desigualdade (5.13) é válida pelo lema 5.3.2. □

Para obter um algoritmo polinomial que é $8 \log n$-aproximado para o problema UML precisamos apresentar um algoritmo $\mathcal{A}_{\mathrm{SC}}$ que seja $4 \log n$-aproximado para o problema de cobertura por conjuntos e que não precise de todas as possíveis estrelas. O algoritmo é baseado em uma aproximação para o problema *Quotient Cut*, que é descrito a seguir.

5.3.2 Um Algoritmo $\mathcal{A}_{\mathrm{SC}}$ polinomial para o problema de cobertura por conjuntos

O conjunto de todas as possíveis estrelas e, por conseqüência, a coleção de conjuntos S_{SC} da linha 2 do algoritmo possui tamanho exponencial e a geração de todos os conjuntos tornaria nosso algoritmo inviável. Nesta seção mostraremos como implementar o algoritmo guloso para o problema de cobertura por conjuntos sem gerar todos os possíveis conjuntos.

Os passos de 1 a 4 do algoritmo GUL, que dizem respeito a transformação de uma instância do problema de classificação métrica em uma instância do problema de cobertura por conjuntos, através da criação de todas as estrelas possíveis, devem ser substituídos por um algoritmo que encontra as estrelas de menor custo amortizado que serão usadas no problema de cobertura por conjuntos. Este novo algoritmo basicamente gera um grafo G_l para cada possível classe $l \in L$ e obtém um conjunto U_S de menor custo amortizado referente à classe l.

Em uma dada iteração para encontrar o conjunto U_S que minimize $w_S/|U_S \cap U|$, onde U é o conjunto dos objetos não cobertos, é construído um grafo completo G_l da seguinte forma:

a) O conjunto dos vértices $V(G_l) = P \cup \{l\}$, onde P é o conjunto dos objetos e l é uma classe de L;

b) O peso $w_{G_l}(u, v)$ das arestas $e = \{u, v\}$ no grafo G_l, para $e = \{u, v\} \in U$, é igual a w_e e o peso $w_{G_l}(u, l)$ para as arestas (u, l), onde $u \in U$ é igual a c_{ul}.

Em outras palavras, o custo de uma aresta entre uma classe e um objeto é o custo de associação e o custo de uma aresta entre dois objetos é o custo de separação.

Lema 5.3.4 *Se $C \subseteq P$ é um corte de G_l, $l \notin C$, o custo do corte C, $c(C) = \sum_{u \in C, v \notin C} w_{G_l}(u, v)$, é igual ao custo da estrela $S = (l, C)$ para o problema de cobertura por conjuntos.*

Prova.

O lema pode ser provado por contagem. No problema de cobertura por conjuntos, w_S é igual a

$$\sum_{u \in U_S} c_{ul} + \sum_{u \in U_S, v \in P \setminus U_S} w_{uv}, \qquad \text{onde } S = (l, U_S),$$

que é igual ao custo do corte C. Veja a figura 5.12.

□

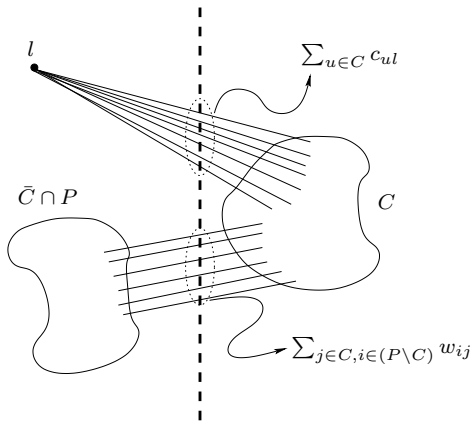

Figura 5.12: Exemplo de um corte C que tem o mesmo custo w'_S para $U_S = C$.

Vimos que o custo do corte no grafo G_l equivale ao custo da estrela S. Gostaríamos que a razão entre o custo do corte C e o número de elementos do corte C fosse mínima. Isto pode ser resolvido pelo problema *Quotient Cut* [18], definido a seguir.

Definição 5.3.5 QUOTIENT CUT PROBLEM (QCP)*: Dado um grafo G com peso $w_e \in R^+$ para cada aresta $e \in E(G)$ e peso $c(v) \in \mathbb{Z}^+$ para cada vértice $v \in V(G)$. O objetivo é encontrar um corte C que minimize $w(C)/\min\{\pi(C), \pi(\bar{C})\}$, onde $w(C) := \sum_{u \in C, v \notin C} w_{uv}$ e $\pi(C) := \sum_{v \in C} c(v)$.*

Se definirmos uma função de peso c_l para cada vértice de G_l como

$$c_l(v) = \begin{cases} 1 & \text{se } v \in V(G_l) \cap P \cap U, \\ 0 & \text{se } v \in V(G_l) \cap (P \setminus U), \\ n & \text{se } v \in V(G_l) \cap L, \end{cases}$$

onde U são os objetos não classificados, então um corte $C := \min_{i \in L}\{QC(G_l, w_{G_l}, c_l)\}$, devolvido por um algoritmo QC f-aproximado para o problema *Quotient Cut*, corresponde a um conjunto S, $S = \{l\} \cup C$ que é no máximo f vezes maior que o conjunto com o custo amortizado mínimo. Um algoritmo $\mathcal{A}_f$ pode ser implementado escolhendo este conjunto. Assim, o seguinte resultado é válido pelo teorema 5.1.5.

Teorema 5.3.6 *Se existir um algoritmo f-aproximado para o problema Quotient Cut com tempo de execução $T(n)$, então existe um algoritmo $2f\,H_n$-aproximado para o problema UML, com complexidade de tempo igual a $O(n\,m\,T(n))$.*

O problema *Quotient Cut* é NP-difícil e Aumann e Rabani [1] apresentam um algoritmo polinomial $O(\log n)$-aproximado para o problema. Mais recentemente Freivalds [10] apresentou uma algoritmo com fator de aproximação 4 para o problema *Quotient Cut*. Em experimentos computacionais efetuados por Freivalds [10], seu algoritmo apresentou um consumo de tempo estimado em $O(n^{1.6})$ quando o grau de cada vértice é uma constante pequena e $O(n^{2.6})$ para grafos densos. Apesar disso, não há uma prova de que sua complexidade de tempo no pior caso seja polinomial. Utilizando o resultado de Freivalds [10] podemos enunciar o seguinte teorema.

Teorema 5.3.7 *Há um algoritmo $8 \log n$-aproximado para o problema UML com complexidade de tempo estimada em $O(m\,n^{3.6})$.*

Note que se I é uma instância para o problema UML, seu tamanho, denotado por $size(I)$, é $O(n(n+m))$. Assim, a complexidade estimada para nosso algoritmo é $O(size(I)^{2.3})$.

Se utilizarmos o algoritmo $O(\log n)$-aproximado de Aumann e Rabani [1] para o problema *Quotient Cut*, podemos substituir o teorema 5.3.8 pelo seguinte teorema.

Teorema 5.3.8 *Há um algoritmo $O(\log^2 n)$-aproximado para o problema UML com complexidade de tempo polinomial.*

5.4 O Fator de aproximação é $\Omega(\log n)$

Nesta seção mostraremos que o fator de aproximação do algoritmo GUL é $\Omega(\log n)$ através da apresentação de uma família de instâncias, onde o fator de aproximação é $\Omega(H_n)$ quando utilizado o algoritmo GUL. Para tanto, assumimos que $\mathcal{A}_{\text{SC}}$ é um algoritmo guloso que seleciona o conjunto de menor custo efetivo em cada iteração.

Dado um inteiro n e valores positivos ϵ_1 e ϵ_2, onde $0 < \epsilon_1 \ll \epsilon_2 \ll 1/n^2$, definimos uma instância $\mathcal{I}_n = (L, P, c, w)$ da seguinte forma:

- $L = \{1, \ldots, n\}$,
- $P = \{1, \ldots, n\}$,
- para cada $i \in P$ e $l \in L$, defina c_{il} como segue

$$c_{il} = \begin{cases} \epsilon_1 & \text{se } i \in \{1, \ldots, n-1\} \text{ e } l = i, \\ \epsilon_2 & \text{se } i \in \{1, \ldots, n-1\} \text{ e } l = n, \\ 1 & \text{se } i = n \text{ e } l = n, \\ \infty & \text{demais casos,} \end{cases}$$

- para cada $i, j \in P$, defina w_{ij} como segue

$$w_{ij} = \begin{cases} \dfrac{1}{n-i+1} & \text{se } i \in \{1, \ldots, n-1\} \text{ e } j = n, \\ 0 & \text{caso contrário.} \end{cases}$$

A instância $\mathcal{I}_n$ é apresentada na figura 5.13. No próximo teorema mostraremos que o algoritmo GUL pode obter uma solução para a instância $\mathcal{I}_n$ com um fator de $\Omega(\log n)$ do ótimo.

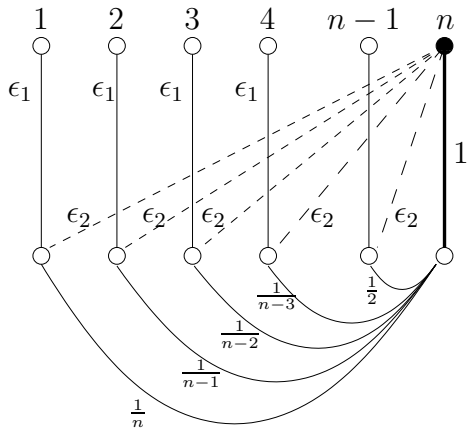

Figura 5.13: Uma instância $\mathcal{I}_n$.

Teorema 5.4.1 *Se o algoritmo* GUL *usa um algoritmo guloso como subrotina para o problema de cobertura por conjuntos, então* $\frac{\text{GUL}(\mathcal{I}_n)}{\text{OPT}(\mathcal{I}_n)} \to H_n$ *com* $\epsilon_1, \epsilon_2 \to 0$*, onde* $\text{OPT}(\mathcal{I}_n)$ *é o valor de uma solução ótima para* $\mathcal{I}_n$*.*

Prova. A solução ótima para a instância $\mathcal{I}_n$ é a estrela (n, P) com custo $1 + \epsilon_2(n-1)$. Agora, vamos provar que a solução obtida pelo algoritmo GUL pode ser formada pelas estrelas $(1, \{1\}), (2, \{2\}), \ldots, (n, \{n\})$ e custar $H_n + \epsilon_1(n-1)$.

O teorema pode ser provado por indução no número de iterações.

Dado uma estrela $S = (l, U_S)$, para $l \in L$ e $U_S \subseteq P$, denotamos por $W_i(l, U_S)$ o custo amortizado de S na i-ésima iteração, e seu valor é $W_i(l, U_S) = \frac{w'_S}{|U_S \cap U|}$, onde U é o conjunto de objetos não associados até a i-ésima iteração.

Apesar do número de estrelas ser grande, podemos considerar apenas algumas estrelas, como é mostrado no próximo fato.

Fato 5.4.2 *Todas as estrelas que possuem custo limitado são da seguinte forma*

- $(i, \{i\})$ *para* $i = 1, \ldots, n$*;*
- (n, U_S) *para qualquer conjunto* $U_S \subseteq P, U_S \neq \emptyset$*.*

Considere a primeira iteração do algoritmo. Neste caso vamos provar que a estrela de menor custo efetivo é a estrela $(1, \{1\})$. A seguinte desigualdade é imediata:

$$W_1(1, \{1\}) < W_1(2, \{2\}) < \ldots < W_1(n, \{n\}). \tag{5.14}$$

$$W_1(i, \{i\}) < W_1(n, \{i\}), \text{ para todo } i \in \{1, \ldots, n-1\}. \tag{5.15}$$

Da desigualdade (5.14) e (5.15) podemos concluir que $W_1(1, \{1\})$ é menor ou igual ao custo efetivo de todas as estrelas de tamanho 1.

Como $W_1(1, \{1\}) = 1/n + \epsilon_1$ e $W_1(n, P) = 1/n + \frac{n-1}{n}\epsilon_2$, temos

$$W_1(1, \{1\}) < W_1(n, P). \tag{5.16}$$

Considere as estrelas que possuem o conjunto de objetos Q, onde $n \in Q$ e $Q \subseteq P$. Como a adição de um objeto $i \in P \setminus Q$ em Q diminui o custo de separação e o aumento no custo de associação não é significativo (apenas ϵ_2 é acrescentado), temos

$$W_1(n, P) \leq W_1(n, Q). \tag{5.17}$$

Através da desigualdade (5.16) e (5.17) podemos concluir que qualquer estrela que contenha o objeto n tem custo maior que $W_1(1, \{1\})$.

Agora, considere uma estrela com o conjunto de objetos $Q \subseteq P \setminus \{n\}, Q \neq \emptyset$. Um argumento minimal diz que

$$W_1(n, \{i\}) \leq W_1(n, Q), \text{ para } i = \min\{Q \cap U\}. \tag{5.18}$$

A desigualdade (5.18) é válida devido à

$$W_1(n, \{i\}) = \min_{i' \in Q \cap U} \{\frac{1}{n - i' + 1}\} + \epsilon_2$$

e

$$W_1(n, Q) \geq \text{average}_{i' \in Q \cap U} \{\frac{1}{n - i' + 1}\} + \epsilon_2.$$

Através das desigualdades anteriores podemos concluir que todas as estrelas tem um custo efetivo maior que $W_1(1, \{1\})$. Assim, a estrela $(1, \{1\})$ entra na solução obtida pelo algoritmo GUL na primeira iteração.

Atualizando U, o conjunto de objetos não associados, podemos notar uma propriedade que é invariante. Na segunda iteração esta claro que

$$W_2(2, \{2\}) < W_2(3, \{3\}) < \ldots < W_2(n, \{n\}) \tag{5.19}$$

e como $W_2(2, \{2\}) = 1/(n-1) + \epsilon_1$ e $W_2(n, P) = 1/(n-1) + \frac{n-1}{n-1}\epsilon_2$ temos

$$W_2(2, \{2\}) < W_2(n, P). \tag{5.20}$$

De modo similar podemos concluir que todas as estrelas tem um custo efetivo maior que $W_2(2, \{2\})$ na segunda iteração. Assim, o algoritmo guloso selecionará a estrela $(2, \{2\})$ para entrar na solução.

O posso da indução é direto. Por conseqüência, a solução produzida pelo algoritmo GUL consiste de $\{\phi(1) = 1, \phi(2) = 2, \ldots, \phi(n) = n\}$. □

Capítulo 6

Resultados

Neste capítulo apresentamos uma comparação teórica entre os algoritmos, analisando a complexidade computacional dos algoritmos apresentados para o problema de classificação métrica uniforme. Apresentamos, também, os resultados obtidos quando estes algoritmos são aplicados em instâncias geradas computacionalmente e instâncias práticas.

6.1 Experimentos Computacionais

Na literatura encontramos três principais abordagens para o problema de classificação métrica, que são:

- a abordagem apresentada por Kleinberg e Tardos [17] (seção 3.1 pg.13), denotada por M_{KT}. Desta abordagem temos dois algoritmos, um para o problema UML e um para o problema ML, denotaremos por A_K o algoritmo para o problema UML;
- a abordagem apresentada por Chekuri et al.[7], denotada por M_{CKNZ}. Desta abordagem temos algoritmos para os problemas ML, UML, LML e TML, denotaremos por A_C o algoritmo para o problema UML;
- a abordagem de Gupta e Tardos [11], denotada por M_{GT}, da qual temos um algoritmo para o problema TML, que denotaremos por A_{GT}.

As abordagens de M_{KT} e M_{CKNZ} são baseadas na resolução de um programa linear. Os algoritmos A_K, baseados na abordagem M_{KT}, precisam resolver um programa linear que é composto de $O(n^2m)$ restrições com $O(n^2m)$ variáveis, correspondendo a uma matriz com $O(n^4m^2)$ elementos. Já os algoritmos baseados na abordagem M_{CKNZ}, para uma instância de m classes e n objetos, resolve um programa linear que possui $O(n^2m^2)$ restrições e $O(n^2m^2)$ variáveis, correspondendo assim a uma matriz com $O(n^4m^4)$ elementos.

A abordagem M_{GT} por sua vez não utiliza programação linear e é baseada em um algoritmo de fluxo em redes. O algoritmo A_{GT} executa a cada iteração um algoritmo que busca o corte de menor capacidade em um rede de fluxos, construída sobre o conjunto de objetos e classes, contendo $O(n^2m)$ arestas. O algoritmo do corte mínimo é aplicado $O((m/M)(\log Q_0 + \log \epsilon^-))$ vezes, para conseguir uma aproximação de $(4+\epsilon)Q(f^*)$ vezes o ótimo, onde M é o valor do truncamento da distância entre as classes e Q_0 é o custo da classificação de partida, geralmente uma classificação aleatória. Veja o capítulo 4 para maiores informações.

Implementamos os algoritmos A_K, A_C e o nosso algoritmo, denotado por GUL. A primeira dificuldade que observamos ao iniciar a fase de testes foi a falta de instâncias para testar os algoritmos implementados. Tentamos entrar em contato com outros pesquisadores na busca de instâncias reais para o problema de classificação uniforme, mas não obtivemos sucesso. Devido à necessidade de instâncias para atestar a correção dos algoritmos e comparar seus resultados, construímos instâncias computacionalmente. Na subseção seguinte apresentamos uma análise das instâncias utilizadas no processo.

6.1.1 As Instâncias

Em nosso algoritmo, a cada iteração, ocorre indiretamente a construção de uma estrela de menor custo amortizado aproximada para todas as classes. Para simplificar a análise das instâncias, vamos considerar a construção das estrelas de menor custo amortizado ótimas.

Ao construirmos uma estrela de menor custo amortizado para uma classe i com um conjunto P de n objetos, um subconjunto de objetos $P' \subseteq P$ é escolhido para formar a estrela da classe i. Os objetos em P', que formarão a estrela para a classe i, são aqueles que fazem com que o valor da equação

$$\frac{\sum_{u \in P'} c_{ui} + \sum_{u \neq v : u \in P', v \notin P'} w(u,v)}{|P'|}$$

seja minimizado.

Analisando o nosso algoritmo percebemos que o tempo necessário para a obtenção de uma solução depende do número de objetos escolhidos para compor a estrela em cada iteração. Assim, a maior complexidade de tempo para o algoritmo GUL ocorre quando o número de objetos associado a cada iteração é pequeno, tendo em vista que a cada construção de uma estrela de menor custo efetivo é executado o algoritmo para o problema *Quotient Cut*, que possui complexidade de tempo igual a $O(n^{2,6})$.

A escolha dos objetos que vão compor a estrela é baseada nos seus custos de associação e separação. O nosso objetivo é construir instâncias que demandem maior tempo de execução, para que o algoritmo GUL não seja favorecido no momento da comparação. Para construir tais instâncias temos que escolher quais valores serão atribuídos às funções do custo de associação e do custo de separação. Para que o número de objetos escolhidos a cada iteração seja pequeno,

preferencialmente um, o custo de uma estrela, que contenha apenas um objeto u, tem que ser menor que o custo amortizado de qualquer outra estrela que contém mais de um objeto.

O primeiro passo para obter uma estrela que seja composta por apenas um objeto u é definir o custo de separação menor que o custo de associação. Considerando que o grafo $G = (P, E)$ é completo, ou seja, um objeto relaciona-se com todos os outros, o valor do custo de associação tem que ser proporcional ao número de objetos e ao custo de separação para que $c_{ui} \geq \sum_{v \neq u : v \in P} w(u, v)$.

Com essa configuração o objeto u é escolhido somente se $c_{ui} = \min_{v \in P} c_{vi}$. Como todos os custos de associação são grandes em relação aos custos de separação, e o valor de c_{ui} é o menor de todos, então um novo objeto v será acrescentado à estrela somente se o custo amortizado para u e v for menor que o custo da estrela construída somente com o objeto u. Ou seja,

$$\begin{aligned}
\frac{c_{ui} + \sum\limits_{v \neq u : v \in P} w(u, v)}{1} &< \frac{c_{ui} + c_{vi} + \sum\limits_{y \in P \setminus \{u,v\}} w(y, u) + \sum\limits_{y \in P \setminus \{u,v\}} w(y, v)}{2} \\
&= \frac{2 \cdot c_{ui} + (c_{vi} - c_{ui}) - w(u, v) + 2 \sum\limits_{y \neq u : y \in P} w(u, y)}{2} \\
&= c_{ui} + \sum_{y \neq u : y \in P} w(u, y) + \frac{(c_{vi} - c_{ui}) - w(u, v)}{2}.
\end{aligned}$$

Para que apenas o objeto u faça parte da estrela da classe i, ou seja, para que a desigualdade seja válida, o valor de $c_{vi} - c_{ui}$ tem que ser maior que $w(u, v)$. Baseados nisso, se definirmos o custo de associação c_{ui}, para todo $u \in P$ e $i \in L$, igual a $random(100 \cdot n \cdot c)$, onde $n = |P|$, c é uma constante e $random(t)$ devolve um valor aleatório entre 0 e t, e $w(u, v) = random(c)$, teremos que o valor esperado de instâncias geradas que vão dar origem a estrelas com mais de um objeto é pequeno. Alguns testes computacionais mostram que este número é em torno de 1%.

6.1.2 Os Testes e Seus Resultados

Implementamos o algoritmo para o problema de classificação métrica uniforme de Kleinberg e Tardos [17], denotado por A_K, o algoritmo apresentado por Chekuri et al. [7], denotado por A_C, e o nosso algoritmo, denotado por GUL. Como visto anteriormente, os algoritmos A_K e A_C utilizam programação linear para obter uma solução fracionária para o problema. Nestes casos utilizamos o pacote de programação linear Xpress-MP [19]. A implementação foi feita em linguagem $C++$. Todos os testes foram executados em uma Athlon XP com 1,2 Ghz, 700 MB de RAM.

Executamos cinco baterias de testes. Nas instâncias utilizadas na primeira bateria de teste, o número m de classes é igual à metade do número n de objetos, ou seja, $(n, m) \leftarrow (\lfloor \frac{2k}{3} \rfloor, \lceil \frac{k}{3} \rceil)$. Outra característica destas instâncias são os custos de associação e separação, definidos como $c_{ui} \leftarrow random(1000), \forall u \in P, \forall i \in L$ e $w_e = random(10), \forall e = \{u, v\} : u, v \in P$.

Utilizando o algoritmo A_C a maior instância que conseguimos resolver tem 46 objetos e 24 classes e foi preciso para isso 4 horas e 38 minutos. Não foi possível resolver instâncias maiores que esta, devido ao tamanho muito grande do programa linear gerado. Quando aplicamos o algoritmo GUL na mesma instância, obtivemos uma solução que é 0,25% pior em apenas 7 segundos. A maior instância resolvida pelo algoritmo A_K nesta bateria de testes foi de 80 objetos e 40 classes com tempo de execução de aproximadamente 4 horas e 19 minutos. Quando aplicamos o algoritmo GUL na mesma instância, obtivemos um resultado 1,01% pior em 1,21 minutos.

Nesta primeira bateria de testes percebemos que a maior limitação para o uso do algoritmo A_K é o tempo de execução, enquanto que para o algoritmo A_C, além da restrição do tempo de execução, há também limitações quanto à quantidade de memória utilizada. O tempo gasto pelos algoritmos A_K e A_C foi basicamente para resolver o programa linear correspondente. Veja a figura 6.1 para comparar o tempo de execução de cada algoritmo.

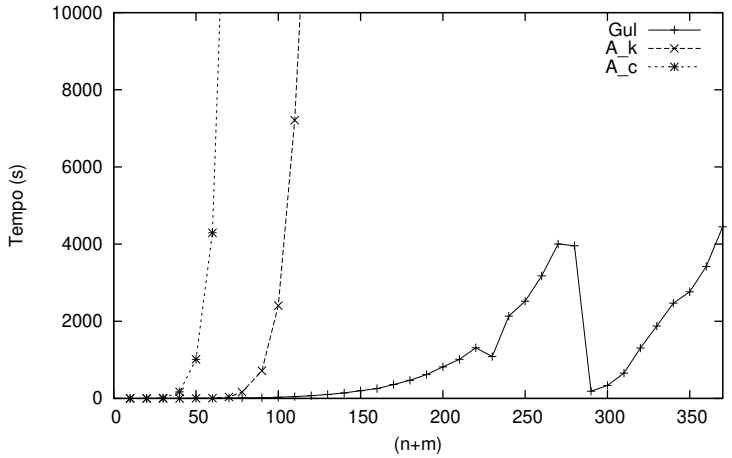

Figura 6.1: Comparação do tempo de execução dos algoritmos implementados na primeira bateria de testes.

Podemos observar na figura 6.1 que há uma redução brusca na função de tempo associada ao algoritmo GUL quando $n+m$ alcança 291. Como citado na subseção 6.1.1, este comportamento ocorre porque nas instâncias de tamanho $n + m < 291$ o número de objetos classificados a cada iteração é baixo, devido ao custo de associação ser, para um objeto ou vários objetos, ser maior que o custo da separação.

Analisando as instâncias utilizadas nesta bateria de testes, temos que o valor esperado do custo de associar um objeto a uma classe é 500. Já o custo esperado da relação entre os objetos

é 5. Assim, para uma instância onde $n + m = 291$, ou seja, $n = 194$ e $m = 97$, o custo esperado de separar um objeto dos demais é $5 * 193 = 965$ que é maior que o custo esperado de associação para esta instância. Isto implica que o custo de classificação é minimizado quando um grande número de objetos são associados à mesma classe em uma mesma iteração. Nesta bateria de testes, quando $n + m < 291$ o número médio de objetos associados em cada iteração foi de 1,2, e para instâncias maiores, o número médio de objetos associados em uma iteração foi 32,3 o que implica na redução do número de iterações e na redução do tempo de execução.

Em uma segunda bateria de testes mantivemos a mesma proporção em relação ao número de classes com o número de objetos e alteramos a relação entre o custo de associação e separação, onde definimos o custo de associação c_{ui}, para todo $u \in P$ e $i \in L$, igual a $random(5 \cdot n)$, onde $n = |P|$, e $w(u, v) = random(5)$. Nestas instâncias não tivemos uma queda brusca no tempo de execução do algoritmo GUL, como ocorreu nas instâncias anteriores, devido ao custo de associação se manter proporcional ao número de objetos e o custo de separação ser constante.

Para essa bateria de testes a maior instância que conseguimos resolver, utilizando o algoritmo A_C, possuía 46 objetos e 24 classes, e foram precisos para isso 5 horas e 12 minutos. Não foi possível resolver instâncias maiores que esta, devido ao tamanho do programa linear gerado ultrapassar 1 Gb de memória. Quando aplicamos o algoritmo GUL na mesma instância, obtivemos uma solução que é 1,25% pior em apenas 7 segundos. A maior instância resolvida pelo algoritmo A_K foi de 120 objetos e 65 classes e o tempo necessário para sua resolução foi de 12 horas e 20 minutos. Quando aplicamos o algoritmo GUL na mesma instância, obtivemos um resultado 0,48% pior em 1,08 minutos. A maior taxa de erro obtida pelo algoritmo GUL em relação ao A_K nessa bateria de testes foi de 11,53%, e em todas as instâncias dessa bateria de testes, quando aplicado o algoritmo GUL, havia um número pequeno de objetos associados em cada iteração. Na média 1,03 objetos associados a cada iteração. Veja a figura 6.2 com o gráfico de comparação dos resultados obtidos.

Em uma terceira bateria de testes buscamos variações na relação entre o número de objetos e o número de classes, fixando n ou m e variando o outro. Nesta fase dos testes não utilizamos o algoritmo A_C por este possuir alto custo computacional, e as suas soluções serem iguais às soluções apresentadas pelo algoritmo A_K. Os resultados estão apresentados na tabela 6.1. A coluna Erro(%) corresponde ao valor de $(\text{GUL}(I)/A_K(I) - 1) \cdot 100$, para cada instância I. A maior taxa de erro obtida neste caso é de 1,25% e, como esperado, o ganho de tempo é considerável quando fixamos o número de classes e aumentamos o número de objetos.

Podemos observar nos resultados da terceira bateria de testes, ilustrados na tabela 6.1, que o algoritmo A_K teve um acréscimo mais significativo no seu tempo de execução quando fixamos o número de classes e aumentamos o número de objetos. Este comportamento já era esperado, tendo em vista que o tempo do algoritmo A_K é baseado na resolução de um programa linear com $O(n^2m)$ restrições e $O(n^2m)$ variáveis. O mesmo ocorre com o algoritmo GUL mas em uma escala menor, pelo fato de que a complexidade do algoritmo GUL é $O(m\, n^{3,6})$ para grafos

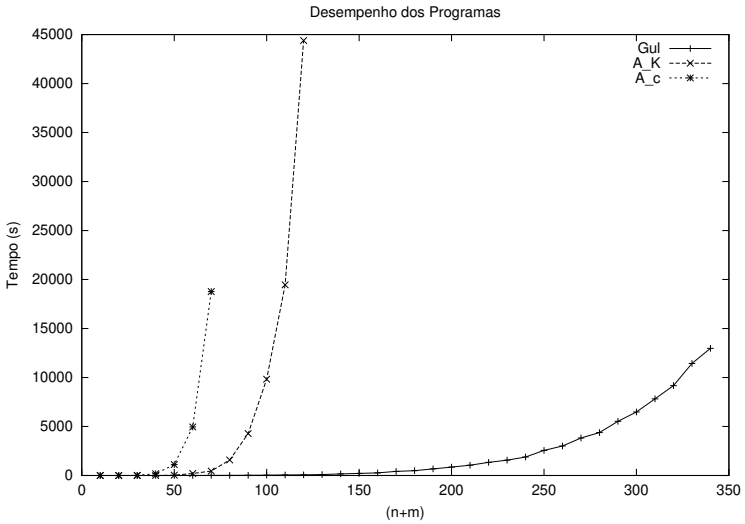

Figura 6.2: Comparação do tempo experimental dos algoritmos implementados.

densos.

Uma outra bateria de testes foi executada sobre instâncias com número fixo de objetos e de classes, 40 objetos e 20 classes, e variando a relação entre o custo de associação e o custo de separação. O custo de associação é dado por $c_{ui} \leftarrow random(1000)$, e o custo de separação é dado por $w(u, v) \leftarrow random(10\rho)$, onde ρ é um valor inteiro e varia de 1 até 100. Os resultados podem ser visto na tabela 6.2.

Visto que o custo de separação aumenta quando ρ aumenta, grandes estrelas se tornam promissorras para fazer parte da solução. Quando grandes estrelas entram na solução, o número de iterações decresce. Note que a instância mais difícil ocorre na transição entre grandes estrelas e pequenas estrelas. Quando o custo de separação é grande ($\rho > 80$), o problema se torna fácil, já que o objetivo se torna conectar todos os objetos em uma classe. Quando $\rho = 80$ a instância correspondente é "difícil" para o algoritmo GUL, já que o fator de aproximação aumenta, esta instância também é "difícil" para o algoritmo A_K, já que o tempo de execução também aumenta. Deste experimentos podemos concluir que a razão entre o custo de associação e separação é uma característica importante em instâncias difíceis.

Observamos que menos de 1% das instâncias utilizadas até o momento da execução desta bateria de testes tinham gerado soluções fracionárias para os programas lineares (KT), de Kleinberg e Tardos [17], e (CKNZ), de Chekuri *et al.* [7]. Nesta última bateria de testes surgiram algumas instâncias que geraram soluções fracionárias para os programas lineares, e após analisarmos estas instâncias conseguimos identificar características que nos levaram a uma nova bateria de testes, onde cerca de 10% das instâncias geradas resultaram em soluções fracionárias para os programas lineares. Como características destas instâncias temos que o tamanho do conjunto de objetos é igual a 40, o número de classes é igual a 20, a relação w_{uv} entre os ob-

jetos é igual a $random(8)$ e o custo de associação é igual a $random(192)$ ou $random(200)$ ou $random(208)$.

O maior erro entre o valor devolvido pelo algoritmo A_K e a solução do programa linear (KT), para as instâncias com as características apresentadas anteriormente, foi de 24,17% e foram necessários 17 minutos para obter a solução neste caso, enquanto que o erro para o algoritmo GUL para esta mesma instância foi de 7,15% usando apenas 4 segundos. O erro médio das instâncias cujo valor da solução do programa linear é fracionária é de 7,02% com um tempo de execução médio de 16,93 minutos, enquanto que o erro médio para o algoritmo GUL quando aplicado nas mesmas instâncias é de 5,8% com tempo de execução médio de 3,02 segundos.

Para todas as instâncias utilizadas nos testes anteriores o grafo $G = (P, E)$ é completo, ou seja, cada objeto se relaciona com todos os outros. Baseados nas características das instâncias da última bateria de testes, geramos instâncias onde o grafo $G = (P, E)$ é um grafo não muito denso, com grau médio dos vértices igual a 2. Mais precisamente, estas novas instâncias são formadas por 20 objetos e 10 classes, com em média 40 arestas entre os objetos, custo de associação é igual a $random(20)$ e o custo de separação é igual a $random(8)$. Cerca de 8% das instâncias com estas características geravam resultados fracionários para os programas lineares e o maior erro para o algoritmo A_K em relação à solução fracionária do programa linear foi de 21,34% com erro médio de 6,47%, enquanto que o maior erro para o algoritmo GUL, quando comparado com a solução fracionária do programa linear, é de 50,89% e com erro médio de 13,46%.

Os resultados computacionais que obtivemos com estas instâncias nos forneceu maior conhecimento para gerar instâncias difíceis. Isto nos conduziu às instâncias $\mathcal{I}_n$ que provam que o algoritmo GUL possui um fator de aproximação de $\Omega(\log n)$ (veja a seção 5.4).

Para ilustrar a aplicabilidade do algoritmo primal-dual em instâncias práticas, aplicamos o algoritmo ao problema de recuperação de imagens, com uma imagem degenerada por ruídos. A imagem possui uma malha de pontos em tons de cinza e dimensões 60x60, com um total de 3600 pontos (objetos) para serem classificados nas cores branco ou preto. Para definir o custo de associação e separação consideramos que cada cor é um inteiro entre 0 e 255. O custo de associação de um objeto u a uma classe i é dado por $c_{ui} = |clr(u) - clr(i)|$, onde $clr(u)$ é a cor atual do ponto u e $clr(i)$ é a cor associada à classe i. O custo de separação de um objeto u e um objeto v é dado por $w_{uv} = 255 - |clr(u) - clr(v)|$. Nestas imagens consideramos apenas os oito vizinhos mais próximos de cada ponto.

As seguintes imagens, apresentadas nas figuras 6.3–6.6, apresentam o resultado obtido aplicando o algoritmo primal-dual. Em cada figura, a imagem da esquerda foi obtida inserindo alguns ruídos e a imagem da direita é a imagem obtida após a classificação dos objetos.

A diferença de tempo de classificação entre as imagens ocorre porque o custo de separação nas imagens com menos ruído é maior que em imagens com mais ruído, fazendo com que o

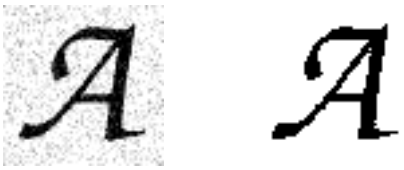

Figura 6.3: Imagem com 25% de ruído. Tempo necessário de 4,8 minutos.

Figura 6.4: Imagem com 50% de ruído. Tempo necessário de 5,32 minutos.

Figura 6.5: Imagem com 75% de ruído. Tempo necessário de 8,51 minutos.

Figura 6.6: Imagem com 100% de ruído. Tempo necessário de 12,22 minutos.

número de objetos classificados a cada iteração seja maior.

Apesar destes tempos serem altos para aplicações de processamento de imagens, eles servem para ilustrar a aplicabilidade e a qualidade das soluções do nosso algoritmo. Além disso, em instâncias reais, possivelmente grandes, é possível trabalhar com o sistema de janelas, onde a imagem é dividida em partes menores e então é aplicado o algoritmo de classificação, fazendo com que o tempo de execução seja menor. Claramente, o problema de classificação métrica é mais geral que o problema de restauração de imagens e o algoritmo primal-dual se mostra apropriado para instâncias de tamanho moderado e grande.

Instância		Tempo (s)		
Objetos (n)	Classes (m)	GUL	A_K	Erro (%)
40	5	0	1	0.18
40	10	2	3	0.23
40	15	3	6	0.67
40	20	4	9	0.29
40	25	5	12	0.47
40	30	5	23	0.19
40	35	6	30	0.66
40	40	7	36	0.27
40	45	8	49	**1.25**
40	50	9	56	0.78
40	55	10	75	0.77
40	60	11	94	0.44
40	65	11	68	0.37
5	40	0	1	0
10	40	1	0	0
15	40	0	1	0
20	40	1	2	0.87
25	40	2	3	0.83
30	40	3	7	0.46
35	40	4	17	0.70
45	40	11	77	0.62
50	40	16	131	0.56
55	40	22	400	0.38
60	40	26	912	0.47
65	40	33	2198	0.60

Tabela 6.1: Tempos e soluções para os algoritmos A_K e GUL com variações no número de objetos e classes.

Tempo (s)				Número de
GUL	A_K	Erro (%)	ρ	iterações
3	5	0.34	1	36
4	7	0.39	5	37
4	7	0.57	10	38
4	11	0.75	15	39
4	11	0.43	20	38
3	25	0.66	25	36
3	21	0.47	30	37
4	23	0.51	35	38
3	35	0.51	40	37
4	41	0.88	45	36
4	67	0.68	50	36
4	114	1.06	55	36
4	168	1.75	60	36
4	203	1.26	65	32
4	439	2.26	70	32
4	831	4.32	75	29
4	1182	13.33	80	23
2	790	0	85	1
0	549	0	90	1
0	262	0	95	1
0	114	0	100	1

Tabela 6.2: Tempos e soluções para os algoritmos A_K e GUL com variações na relação entre o custo de associação e o custo de separação.

Capítulo 7

Considerações Finais

Após estudos que culminaram nesta dissertação, verificamos que os problemas de classificação métrica ainda merecem mais atenção da comunidade científica. Os algoritmos de aproximação existentes ainda apresentam complexidade de tempo alta, fazendo com que sua aplicabilidade seja restrita, como é o caso dos algoritmos que têm como passo inicial a resolução de um programa linear grande. Motivados pelo alto custo computacional dos algoritmos presentes na literatura e através do estudo de problemas correlatos, como o problema de localização de recursos e o problema de cobertura por conjuntos, desenvolvemos um novo algoritmo para o problema de classificação uniforme.

Após finalizarmos a fase de implementação dos algoritmos, começamos a busca por instâncias reais para o problema, mas não obtivemos sucesso. A solução encontrada foi gerar instâncias computacionalmente e também oriundas do problema de recuperação de imagens degeneradas por ruídos. Utilizando as instâncias geradas computacionalmente comparamos os resultados do nosso algoritmo com os resultados dos algoritmos baseados em programação linear.

Apesar do novo algoritmo poder ser implementado em tempo polinomial com fator de aproximação de $O(\log^2 n)$, optamos por implementar uma versão, que dá um fator de aproximação de $8 \log n$, mas não possui complexidade de tempo comprovadamente polinomial. Por outro lado ele se mostrou rápido e apresentou bons resultados quando comparado com os algoritmos baseados em programação linear. A qualidade das soluções obtidas, apesar da análise apontar um algoritmo $8 \log n$-aproximado, são boas, tendo uma taxa de erro média inferior a 14% da solução ótima e o maior erro encontrado foi de 50.89% em relação ao ótimo fracionário do programa linear. Por ter tempo de execução menor e usar menos memória, o novo algoritmo obtém soluções para instâncias de tamanho moderado e grande, enquanto que os algoritmos baseados em programação linear resolvem apenas instâncias de pequeno porte.

Sabemos que as instâncias utilizadas nos testes não refletem a realidade mas elas serviram para o nosso propósito de comparar a eficácia do nosso algoritmo em relação aos demais.

7.1 Trabalhos Futuros

Apresentamos a seguir alguns pontos de pesquisas que consideramos promissores:

- Estudar a aplicação da técnica de programação semidefinida ao problema de classificação métrica.
- Buscar instâncias práticas e interessantes para o problema, para que possamos analisar melhor o funcionamento do algoritmo em instâncias reais.
- Continuar na busca por um algoritmo primal-dual para o problema de classificação que apresente um fator de aproximação melhor que $O(\log n)$.
- Acrescentar um custo de utilização para cada classe, que deverá ser pago quando, pelo menos, um dos objetos for associado àquela classe. Observe que com esta alteração temos um problema muito semelhante ao problema de localização de recursos, com uma nova restrição que dois clientes fortemente relacionados devem ser atendidos pelo mesmo recurso.
- Acrescentar a característica de que a relação entre os objetos obedeça alguma métrica mais restrita. Talvez com esta característica o problema de classificação métrica possa ser resolvido mais facilmente através de alguma técnica que leve em consideração esta informação.

Referências Bibliográficas

[1] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

[2] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[3] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annuall ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[4] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. Number TR97-1658, page 10. IEEE Conference on Computer Vision and Pattern Recognition, 3 1997.

[5] M.H. Carvalho, M.R. Cerioli, R. Dahab, P. Feofiloff, C.G. Fernandes, C.E. Ferreira, K.S. Guimarães, F.K. Miyazawa, J.C. Pina Jr., J. Soares, and Y. Wakabayashi. *Uma introdução sucinta a algoritmos de aproximação*. Editora do IMPA - Instituto de Matemática Pura e Aplicada, Rio de Janeiro–RJ, 2001. M.R. Cerioli, P. Feofiloff, C.G. Fernandes, F.K. Miyazawa (editors).

[6] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for group steiner trees and k-median. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 114–123, 1998.

[7] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the ACM Symposium on Discrete Algorithms*, pages 109–118, 2001.

[8] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. of Oper. Res.*, 4(3):233–235, 1979.

[9] J. Fakcheroenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 448–455, 2003.

[10] K. Freivalds. A nondifferentiable optimization approach to ratio-cut partitioning. In *Proc. 2nd Workshop on Efficient and Experimental Algorithms*, number 2647 in Lectures Notes on Computer Science. Springer, Verlag, 2003.

[11] A. Gupta and E. Tardos. Constant factor approximation algorithms for a class of classification problems. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing*, pages 125–131, 1998.

[12] D.S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[13] H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1998.

[14] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of ACM*, 50(6):795–824, 2003.

[15] K. Jain and V.V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. *Journal of the ACM, Vol. 48*, pages 274–296, 2001.

[16] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[17] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Proceedings of the 40th Annuall IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1999.

[18] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46 , No. 6:787–832, Nov. 1999.

[19] Dash Optimization. *Xpress-MP Manual. Release 13*. 2002.

[20] S. Peleg, M. Werman, and H. Rom. A unified approach to the change of resolution: space and gray-level. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, number 11, pages 739–742, 1989.

[21] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[22] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2000.

[23] D.P. Williamson. Lecture notes on approximation algorithms. Technical Report RC 21409, T.J. Watson Research Center (IBM Research Division), Yorktown Heights, New York, 1998.