MAC 5701 - Tópicos em Ciência da Computação
Relatório
Redes Seguras

Aluno: *Antonio João Ferreira Francisco* (`ajoaoff@ime.usp.br`)
Orientador: *Carlos Eduardo Ferreira* (`cef@ime.usp.br`)

Sumário

1 Introdução

Redes seguras são redes em que, para alguns nós importantes, queremos garantia de alta conectividade. Queremos ter vários caminhos entre alguns nós, enquanto alguns outros podem até não estar conectados. Para descobrir a rede de menor custo que mantém essas restrições, a maneira mais usual é a modelagem como o problema de Steiner generalizado, para o qual existem vários algoritmos de aproximação. Outra abordagem, poliédrica, também será objeto deste estudo. Vamos estudar alguns desses algoritmos de aproximação, um pouco da abordagem poliédrica para o problema.

Este problema tem várias aplicações como, por exemplo, vários pontos de um país que precisam ser conectados por redes de fibra ótica, e entre alguns pontos importantes a tolerância a falhas tem que ser maior do que em outros. Outro exemplo é uma rede telefônica, onde os clientes podem escolher quantas conexões entre cada ponto devem existir.

O problema, chamado *Survivable network design problem (SNDP)* consiste em, dados um grafo não-orientado $G = (V, E)$, um custo não-negativo c_e, para cada aresta $e \in E$, e *conectividade mínima* r_{ij}, para cada par de vértices $i, j \in V$, encontrar o subgrafo de custo mínimo para o qual, para cada par de vértices i, j, existem pelo menos r_{ij} caminhos disjuntos nas arestas ligando i a j. Esse problema foi bastante estudado durante a década de 90, e foram descobertos alguns algoritmos de aproximação para ele. Vamos focar no algoritmo descrito em [3].

2 Um algoritmo eficiente para o SNDP

Este algoritmo foi descrito por Gabow, Goemans e Williamson em [3], e é uma melhora do algoritmo em [9]. Os dois algoritmos têm a mesma razão de aproximação, $2k - 1$, onde $k = \max r_{ij}, i, j \in V$, sendo que o primeiro tem complexidade $\mathrm{O}(k^2n^2 + kn^2\sqrt{\log\log n})$ e o segundo tem complexidade $\mathrm{O}(k^3n^4)$.

Daqui pra frente, vamos nos referir ao algoritmo em [3].

O algoritmo utiliza programação inteira, baseado numa função *própria* f. Uma função $f : 2^V \to N$ é chamada própria se:

- for simétrica, isto é, $f(S) = f(V - S)$ para todo $S \subseteq V$;
- for maximal, isto é, se A e B são disjuntos, então $f(A \cup B) \leq \max\{f(A), f(B)\}$.

$f(\emptyset) = 0$.

O SNDP pode ser representado pelo seguinte programa inteiro:

$$\text{(IP)} \qquad \begin{array}{l} \min \sum_{e\in E} c_e x_e \\ \text{sujeito a } x(\delta(S)) \geq f(S), S \subset V, \\ \qquad\qquad x_e \in \{0,1\}, e \in E, \end{array}$$

onde $\delta(S)$ representa as arestas com exatamente uma ponta em S.

O resultado de [3] é o seguinte:

Teorema 1 *Se a função própria f tem apenas l valores diferentes de zero, que chamamos de $0 = \rho_0 < \rho_1 < \cdots < \rho_l$, então existe um algoritmo de aproximação para (IP) com razão de aproximação de pelo menos $2\sum_{i=1}^{l} H(\rho_i - \rho_{i-1})$, onde H é a função harmônica $H(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$. Se $\rho_l = 1$ e $l \geq 2$ então a razão melhora em 1 unidade.*

Vamos denotar daqui pra frente como $f_{\max}$ o maior requerimento, isto é, $f_{\max} = \max_S f(S)$. Assim, a razão de aproximação de acordo com o Teorema 2.1 é no máximo $2f_{\max}$ ou $2f_{\max} - 1$ se $f_{\max} \geq 2$. Agora, vamos descrever o algoritmo, que tem eficiência $O(f_{\max}^2 n^2 + f_{\max} n^2 \sqrt{\log\log n})$.

O algoritmo tem $f_{\max}$ fases. Em cada fase a solução corrente é aumentada. Seja F_{p-1} representar as arestas escolhidas nas primeiras $p-1$ fases, e seja $\delta_A(S) = A \cap \delta(S)$, para algum $A \subseteq E$. Então, na fase p, encontramos um conjunto de arestas $F \subseteq E - F_{p-1}$, tal que, se $f(S) \geq p$ e $|\delta_{F_{p-1}}(S)| = p - 1$, então $|\delta_F(S)| \geq 1$. Então, fazemos $F_p = F_{p-1} \cup F$, mantendo o invariante $|\delta_{F_p}(S)| \geq \min(f(S), p)$, para todos os subconjuntos S. Assim, o conjunto final $F_{f_{\max}}$ é uma solução viável para (IP).

Achar um aumento a cada fase pode ser visto como resolver o seguinte problema:

$$
\begin{array}{lll}
(\mathrm{IP}_h) & & \min \sum_{e \in E_h} c_e x_e \\
& & \text{sujeito a } x(\delta(S)) \geq h(S), S \subset V, \\
& & \qquad\qquad x_e \in \{0, 1\}, e \in E_h,
\end{array}
$$

onde $E_h = E - F_{p-1}$ e $h(S) = 1$ se e somente se $f(S) \geq p$ e $|\delta_{F_{p-1}}(S)| = p - 1$, e $h(S) = 0$ caso contrário.

O algoritmo para cada fase funciona em dois passos. No primeiro, um algoritmo primal-dual constrói um conjunto F de arestas que é solução viável de (IP_h) enquanto também constrói uma solução viável para o dual da relaxação linear de (IP_h). O segundo passo faz uma “limpeza”, removendo arestas desnecessárias de F.

2.1 O algoritmo

Aqui vemos a função principal do algoritmo:

Entrada: Um grafo não orientado $G = (V, E)$, custos $c_e \geq 0$ nas arestas, uma função própria f, e $f_{\max} = \max_S f(S)$
Saída: Um conjunto de arestas $F_{f_{\max}}$ viável para (IP)

$F_0 \leftarrow \emptyset$
For $p \leftarrow 1$ to $f_{\max}$
    Comentário: Inicio da fase p.

$$h_p(S) \leftarrow \begin{cases} 1 & \text{se } f(S) \geq p \text{ e } |\delta_{F_{p-1}}(S)| = p - 1 \\ 0, & \text{caso contrário} \end{cases}$$

    $E_p \leftarrow E - F_{p-1}$
    $F' \leftarrow \mathrm{Uncrossable}(V, E_p, c, h_p)$
    $F_p \leftarrow F_{p-1} \cup F'$

Comentário: Fim da fase p.
Devolva $F_{f_{\max}}$

E agora a função Uncrossable:

Entrada: Um grafo não orientado $G = (V, E)$, custos $c_e \geq 0$ nas arestas e uma função não-cruzável h
Saída: Um conjunto de arestas F' viável para (IP_h)

$F \leftarrow \emptyset$
$i \leftarrow 0$
$d(v) \leftarrow 0$ para todo $v \in V$
Comentário: Implicitamente fazendo $y_S \leftarrow 0$ para todo $S \subset V$
$C \leftarrow$ todos os conjuntos ativos C_a (violados minimais)
Enquanto $|C| > 0$
    $i \leftarrow i + 1$
    Comentário: Início da iteração i
    Comentário: Passo de seleção de aresta
    Para todo $v \in C_a \in C$, incremente $d(v)$ uniformemente de ε até que alguma aresta $e_i = (u, v) \in E_h$ satisfaça $d(u) + d(v) = c_{uv}$ para todo $e_i \in \delta(C_a)$ de algum $C_a \in C$
    Comentário: Implicitamente fazendo $y_{C_a} \leftarrow y_{C_a} + \varepsilon$, para todo $C_a \in C$
    $F \leftarrow F \cup \{e_i\}$
    Comentário: Passo de adição de aresta
    Atualize C
    Comentário: Fim da iteração i
Comentário: Passo de limpeza
$F' \leftarrow F$
Faça todos os cojuntos que foram ativos durante esta fase serem desmarcados. Marque V
Para todo $j \leftarrow i$ até 1
    Comentário: $C_a(e), A(e)$ e arestas especiais serão definidas adiante no texto
    Se e_j é especial e $C_a(e_j)$ não está marcado e $\delta_{F'}(A(e_j)) \supseteq \delta_{F'}(C_a(e_j))$ então
        $F' \leftarrow F' - \{e_j\}$
        Marque $C_a(e_j)$
Devolva F'

O algoritmo tem $f_{\max}$ fases indexadas por $p = 1, \cdots, f_{\max}$. A fase p produz um conjunto F_p tal que $|\delta_{F_p}(S)| \geq \min(f(S), p)$. $F_{f_{\max}}$ é viável para (IP). Na fase p, chamamos a função Uncrossable com o conjunto de arestas $E_h = E - F_{p-1}$ e com a função $h(S)$. Essa função é não-cruzável, isto é, se $h(A) = h(B) = 1$, então ou $h(A - B) = h(B - A) = 1$, ou $h(A \cup B) = h(A \cap B) = 1$, como demonstrado abaixo:

Observação 1 *Seja f uma função própria, e sejam A, B e C uma partição de V. Então o máximo de $f(A)$, $f(B)$ e $f(C)$ não é unicamente atingido.*

Prova. Seja C o conjunto que atinge o máximo. Então, como f é simétrica, $f(C) = f(V-C)$. Como f é maximal e $V - C = A \cup B$, $f(V) \leq \max\{f(A), f(B)\}$.

Corolário 1 *Seja f uma função própria. Para conjuntos disjuntos A e B, o máximo de $f(A)$, $f(B)$ e $f(A \cap B)$ não é unicamente atingido.*

Lema 1 *Sejam f uma função própria e A e B conjuntos de vértices tais que $f(A) \geq p$ e $f(B) \geq p$. Então, ou $f(A - B) \geq p$ e $f(B - A) \geq p$, ou $f(A \cap B) \geq p$ e $f(A \cup B) \geq p$.*

Prova. Assuma, por exemplo, que $f(A - B) < p$. Então, aplicando o corolário acima para $A - B$, $A \cap B$ e A, vemos que $f(A \cap B) \geq p$. Aplicando o corolário para $A - B$, B e $A \cup B$ temos que $f(A \cup B) \geq p$. Os outros casos são similares.

Teorema 2 *Seja F_{p-1} um conjunto de arestas tal que $|\delta_{F_{p-1}}(S)| \geq g_{p-1}(S)$, para todo $S \subseteq V$, sendo $g_p(S) = \min(f(S), p)$. Seja $h(S) = 1$ se $g_p(S) = p$ e $|\delta_{F_{p-1}}(S)| = p - 1$, e $h(S) = 0$ caso contrário. Então, h é "uncrossable".*

Prova. É fácil ver que $h(V) = 0$, já que $g_p(V) = 0$. Sejam A e B conjuntos de vértices tais que $h(A) = h(B) = 1$. Como g_p é própria, então ou $g_p(A \cup B) = g_p(A \cap B) = p$ ou $g_p(A - B) = g_p(B - A) = p$. Suponha que vale a primeira. Como F_{p-1} satisfaz g_{p-1}, devemos ter $|\delta_{F_{p-1}}(A \cup B)| \geq p - 1$ e $|\delta_{F_{p-1}}(A \cap B)| \geq p - 1$. Mas, pela submodularidade de $|\delta_{F_{p-1}}(S)|$, temos que $|\delta_{F_{p-1}}(A)| + |\delta_{F_{p-1}}(B)| \geq |\delta_{F_{p-1}}(A \cup B)| + |\delta_{F_{p-1}}(A \cap B)|$. Assim, devemos ter $|\delta_{F_{p-1}}(A \cap B)| = |\delta_{F_{p-1}}(A \cup B)| = p - 1$. Então $h(A \cap B) = h(A \cup B) = 1$. O outro caso é idêntico.

A função Uncrossable pode ser dividida em dois passos: inclusão de arestas e limpeza. O primeiro passo de Uncrossable consiste de várias iterações, sendo que cada uma consiste de um passo de seleção de aresta e um passo de adição de aresta. No início, F, solução viável de (IP_h), é vazio. Um conjunto S é dito *violado* se $|\delta_F(S)| < h(S)$; isto é, se $h(S) = 1$ mas $\delta_F(S) = \emptyset$. Um conjunto S é dito *ativo* se ele é violado minimal. O primeiro passo mantém uma família C de conjuntos ativos. Um conjunto S é violado se, na chamada de Uncrossable da fase p, $|\delta_{F_{p-1} \cup F}(S)| = p - 1$ e $f(S) \geq p$.

Lema 2 *Se A e B são conjuntos violados em algum ponto de Uncrossable, então ou $A \cap B$ e $A \cup B$ são violados ou $A - B$ e $B - A$ são violados.*

Prova. Como A e B são violados, sabemos que $h(A) = h(B) = 1$. Pelas propriedades de funções *uncrossable*, ou $h(A \cup B) = h(A \cap B) = 1$ ou $h(A - B) = h(B - A) = 1$. Suponha que a primeira é verdadeira. Sabemos que $|\delta_F(A)| + |\delta_F(B)| \geq |\delta_F(A \cup B)|$. Como $|\delta_F(A)| = |\delta_F(B)| = 0$, então $|\delta_F(A \cup B)| = |\delta_F(A \cap B)| = 0$, e, portanto, $A \cup B$ e $A \cap B$ são violados. No outro caso, o argumento é similar.

Dizemos que A *cruza* B se $A \cap B \neq \emptyset$, mas $A \not\subseteq B$ e $B \not\subseteq A$.

Corolário 2 *Nenhum conjunto violado cruza algum conjunto ativo.*

A cada iteração, uma aresta $e \in E_h$ é selecionada da fronteira de algum conjunto ativo e acrescentada em F. A aresta e pode estar na fronteira de um ou dois conjuntos ativos, sendo assim chamada 1-*aresta* ou 2-*aresta*. Quando e é adicionada a F, os conjuntos ativos em C têm que ser atualizados. O(s) conjunto(s) com e na fronteira não são mais ativos, mas um novo conjunto ativo, com e contido nele, pode ser criado.

Um família laminar de conjuntos, L, é uma família de conjuntos tal que, se $A, B \in L$ e $A \cup B \neq \emptyset$, então ou $A \subseteq B$ ou $B \subseteq A$, ou seja, A e B não se cruzam. Outra conseqüência do Lema 2 é a seguinte:

Corolário 3 *Seja* $\cup C$ *o conjunto de todos os conjuntos ativos em todas as iterações. Então* $\cup C$ *é uma família laminar.*

O algoritmo usa o método primal dual. Em cada passo, é mantida uma solução viável y do dual da relaxação linear de (IP$_h$). O dual é o seguinte:

$$
\begin{array}{lll}
(\mathrm{D}_h) & & \min \sum_{S \subset V} h(S) y_S \\
& & \text{sujeito a } \sum_{S: e \in \delta(S)} y_S \leq c_e, e \in E_h, \\
& & \qquad y_S \geq 0, S \subset V.
\end{array}
$$

No começo, $y = 0$. Em cada iteração, o algoritmo incrementa y_a para todos os conjuntos C_a em C de um valor ϵ, o maior possível que não viola as restrições de (D$_h$). Com isto, a desigualdade de (D$_h$) fica justa para alguma aresta $e \in E_h$ na fronteira de algum conjunto ativo. A aresta é acrescentada em F.

O algoritmo não mantém a variável y, pois precisaria de espaço exponencial. Ao invés disso, mantém uma variável $d(u) = \sum_{S: u \in S} y_S$ para cada $u \in V$. Então, aumentar y_a para cada conjunto ativo $C_a \in C$ equivale a aumentar as variáveis $d(u)$ para $u \in C_a \in C$ pelo maior ϵ possível sem violar $d(u) + d(v) \leq c_{uv}$ para $e = (u, v) \in E_h$, e na fronteira de algum conjunto ativo C_a. Então, sendo $a(u) = 1$ se u está em um conjunto ativo e $a(u) = 0$ caso contrário, temos

$$\epsilon = \min_{(i,j) \in E_h : a(i) + a(j) > 0} \frac{c_{ij} - d(i) - d(j)}{a(i) + a(j)}.$$

Suponhamos, por um momento, que nenhum conjunto ativo novo é criado em Uncrossable. Começamos com a família C de conjuntos ativos. Em cada passo, um ou dois conjuntos deixam de ser violados e, portanto, deixam de ser ativos e saem de C. Então, temos no máximo $|C|$ iterações para adicionar arestas. Seja F o conjunto das arestas adicionadas. F é uma floresta no grafo com cada conjunto ativo contraído para um vértice, e mais alguns vértices. Considere que cada floresta "cresce" quando as arestas de F são adicionadas. Cada árvore da floresta cresce com a adição de uma 1-aresta que tem uma ponta em um conjunto ativo (o novo nó) e outra em um conjunto que já foi ativo (o nó existente na árvore). Note que uma aresta ligando dois nós anteriormente ativos nunca será acrescentada e, portanto, duas árvores nunca são ligadas. Assim, 2-arestas sempre começam uma nova árvore. Uma 1-aresta também pode começar uma nova árvore, se ligar um conjunto ativo a outro que nunca foi ativo. Considere as 1-arestas e tais que as arestas adicionadas depois de e geram uma árvore com exatamente os conjuntos ativos imediatamente antes da adição de e. Essas arestas são chamadas *arestas especiais* e são as que podem ser retiradas pelo processo de limpeza.

Vamos agora generalizar para o caso em que novos conjuntos ativos são criados. Seja $\cup C$ a coleção de todos os conjuntos ativos formados em todas as iterações do algoritmo, mais o conjunto V. Seja T uma árvore baseada em $\cup C$, com um vértice v_a para cada $C_a \in \cup C$. Fazemos v_a ser pai de v_b em T se C_a é o menor conjunto em $\cup C$ que contém C_b propriamente. Seja $D(C_a)$ a coleção dos conjuntos correspondentes aos filhos de v_a em T. $D(C_a)$ é uma classe de equivalência de conjuntos ativos. Para uma aresta e, seja $C_a(e)$ o menor conjunto C_a que contém ambas as pontas de e. O conjunto de todas as arestas de F para as quais $C_a(e) = C_a$ é chamado F_{C_a}. As arestas em F_{C_a} são uma classe de equivalência das arestas em F. O comportamento das arestas F_{C_a} nos conjuntos ativos em $D(C_a)$ é o mesmo que no caso acima em que não há novos conjuntos ativos.

Seja $Ac(e)$ os conjuntos em $D(C_a(e))$ que estão ativos logo antes de a aresta e ser escolhida.

Observação 2 *Para qualquer vértice v_a em T, todas as arestas em F_{C_a} devem ser selecionadas antes de alguma aresta em $\delta(C_a)$.*

Prova. Suponha que não seja verdade. Como nem todas as arestas de F_{C_a} foram selecionadas, algum $C'_a \subset C_a$ deve estar ativo. Mas como uma aresta em $\delta(C_a)$ foi selecionada, C_a nunca se tornará ativo, uma contradição.

Observação 3 *Para a primeira aresta selecionada e em F_{C_a}, $Ac(e) = D(C_a(e))$.*

Prova. Se algum conjunto $S \in D(C_a(e))$ não está ativo, então alguma aresta e' foi selecionada em $\delta(S)$ antes de e. Se $C_a(e') \neq C_a(e)$, então $e' \in \delta(C_a(e))$, o que contraria a observação anterior.

Agora, vamos definir mais formalmente as arestas especias. Um conjunto de arestas H forma uma árvore geradora sobre uma família de conjuntos disjuntos de vértices $\{C_{a_i}\}_{i=1}^{k}$ se cada ponta de cada aresta em H está em um dos C_{a_i} e H forma uma árvore geradora no grafo com cada C_{a_i} contraído para um vértice. Então e é uma aresta especial se é uma 1-aresta, e as arestas adicionadas a $F_{C_a(e)}$ depois de e formam uma árvore geradora nos conjuntos em $Ac(e)$.

O passo de "limpeza" olha as arestas de F na ordem reversa em que foram selecionadas para a adição em F. Seja $A(e)$ a união dos conjuntos em $Ac(e)$; isto é, $A(e) = \bigcup_{C_a \in Ac(e)} C_a$. A "limpeza" remove a aresta e de F se e é especial, nenhuma outra aresta de $F_{C_a(e)}$ foi removida, e todas as arestas restantes de F em $\delta(C_a(e))$ estão também em $\delta(A(e))$. A aresta e não é removida se $C_a(e) = V$. Para saber se alguma aresta de $F_{C_a(e)}$ foi removida, o algoritmo marca $C_a(e)$ se e é removida.

2.2 Corretude

Queremos mostrar que o conjunto F' devolvido por Uncrossable é viável para (IP_h).

Teorema 3 *O conjunto de arestas F' resultante depois do processo de limpeza é uma solução viável de (IP_h).*

Prova. O passo de adição de arestas termina sem conjuntos ativos, e portanto, sem conjuntos violados. Só precisamos provar que o passo de "limpeza" mantém viabilidade. Suponha que não, e que F' é viável, mas $F' - e$ não é, e o processo de "limpeza" remove e de F'. Seja $C_a = C_a(e)$. Pela definição do processo de "limpeza", $C_a \neq V$. Seja S um conjunto violado por $F' - e$. Seja I_e a iteração em que a aresta e foi adicionada. O conjunto S estava violado na iteração I_e por causa da ordem da "limpeza".

Todos os conjuntos em $Ac(e)$ também devem estar violados na iteração I_e, pela definição de $Ac(e)$. Pelo Corolário 2.7, S não cruza nenhum conjunto em $Ac(e)$. Podemos mostrar que ou $A(e) \subseteq S$ ou $A(e) \cap S = \emptyset$: como nenhuma aresta de F_{C_a} foi removida ainda no processo de "limpeza", as arestas em F_{C_a} adicionadas depois de e formam uma árvore geradora na família $Ac(e)$. Portanto, se S cruza $A(e)$ mas não cruza nenhum conjunto em $Ac(e)$, S deveria intersectar uma aresta de F_{C_a} adicionada depois de e, uma contradição ao fato de S ser violado.

Assumimos que $A(e) \subseteq S$: se $A(e) \cap S = \emptyset$, então pela propriedade de simetria de h, $V - S$ também está violado e trocados S por $V - S$. Como $C_a \neq V$, o conjunto C_a também está violado na iteração I_e. Pelo Lema 2.6, ou $C_a - S$ e $S - C_a$ são violados, ou $C_a \cap S$ e $C_a \cup S$ são violados na iteração I_e. Mas $C_a - S$ não contém um conjunto ativo, então não é violado. Então, $C_a \cup S$ é violado na iteração I_e. Como ele não era violado antes de e ser removido, F' contém uma aresta e' com exatamente uma ponta em $V - (C_a \cup S)$. A aresta e' não está na fronteira de S, já que $\delta_{F'}(S) = \{e\}$ e $e \neq e'$. Portanto, a outra ponta de e' está em $C_a - S$. Por outro lado, isso implica $e' \in \delta(C_a(e))$; por outro, isso implica $e' \notin \delta(A(e))$. Mas isso contradiz a remoção de e no processo de "limpeza".

3 Algoritmo *When trees collide*

Algoritmo de Agrawal, Klein e Ravi [4]. Utilizando para o seguinte problema: dados um grafo $G = (V, E)$, com custos c_e nas arestas, com uma coleção $\{R_1, R_2, ..., R_b\}$ de requerimentos. Cada requerimento R_i é um par de vértices terminais $\{s_i, t_i\}$, vértices de G, e um valor r_i, o número de caminhos disjuntos nas arestas entre s_i e t_i. Queremos encontrar um subgrafo de G que atenda a todos os requerimentos de custo mínimo. A diferença em relação ao problema anterior é que agora são permitidas arestas múltiplas. A idéia geral do algoritmo é crescer árvores, que ligam os pares de vértices que tem requerimentos de conexão positivos.

3.1 O algoritmo

O algoritmo constrói árvores disjuntas de busca em largura a partir de todos os terminais s_i. É mantido um contador, chamado *timestep*, que aumenta a cada nível da busca, marcando em qual nível se está. Quando duas ou mais árvore colidem, isto é, alcançam um vértice em comum, ele são unidas em um única árvore, e a busca continua da extremidade da nova árvore formada. Quando todos os terminais s_i de uma árvore tem seu correspondente t_i na mesma árvore, essa árvore é contraída para um único vértice e fica *inativa*. Quanda a árvore ainda está crescendo, ela está *ativa*. O algoritmo termina quando não há mais árvores ativas.

Esse algoritmo é simples, pois só trata do caso sem pesos. No caso de termos pesos nas arestas, os *timesteps* são marcados de outra forma. As arestas são "consumidas" aos poucos até serem completamente consumidas. Uma variável é mantida, para cada aresta, indicando quanto da aresta já foi consumida. Em cada iteração, todas as arestas são acrescentadas de λ. Sejam:

$\lambda_1 = \min_e$ o que resta para e ser consumida,

onde o mínimo é sobre as arestas com uma ponta em uma árvore ativa, e

$\lambda_2 = \min_e$ o que resta para e ser consumida,

onde o mínimo é sobre as arestas com as duas pontas em árvores ativas distintas.

Assim, $\lambda = \min\{\lambda_1, \frac{1}{2}\lambda_2\}$.

Todas as arestas são acrescidas agora, as que tem uma ponta em uma árvore ativa são acrescidas de λ, e as que tem as duas pontas em árvores ativas distintas são acrescidas de 2λ, pois estas crescem dos dois lados. Pela definição de λ, pelo menos uma aresta é totalmente consumida em cada iteração. Portanto, pelo menos uma árvore ganha pelo menos um vértice novo. Assim, o número de iterações é no máximo o número de vértices do grafo, e o algoritmo é polinomial.

3.2 Montando a rede

Agora, precisamos construir uma rede viável para o problema. Isso é feito enquanto as árvores são geradas. Para cada árvore, é mantida uma rede conectando todos os terminais da árvore. Isso é feito começando cada terminal como um rede, e, sempre que há uma união, são utilizados caminhos simples unindo as duas redes.

O problema é que as redes construídas dessa maneira não são estão necessariamente conectadas no grafo original, por causa das contrações. Um caminho com um vértice contraído não é geralmente um caminho no grafo original. Dessa forma, é preciso ter mais cuidado quando unimos redes, incluindo arestas entre nós em árvores inativas. Então, quando há uma união, são adicionados às redes caminhos entre o nó comum que foi achado, v, e as redes correspondentes às árvores que estão sendo unidas.

Esse procedimento não é muito simples, já que o vértice v pode ser um vértice contraído, e é preciso expandi-lo antes. Nos caminhos encontrados de v até as redes, podem existir vértices contraídos também, e estes também precisam ser expandidos.

4 Problema *element conectivity*

Problema abordado por Jain, Mandoiu, Vazirani e Williamson [10]. Este problema se diferencia do SNDP pois não trata somente de caminhos disjuntos nas arestas (EC-SNDP), mas também de caminhos disjuntos nos vértices (VC-SNDP). Aqui, é tratado o ELC-SNDP (*element conectiviy survivable network design problem*), que definimos a seguir: os vértices são divididos entre *terminais* e *vértices de Steiner*. Os vértices de Steiner e as arestas podem "falhar"; são os *elementos*. Os terminais não podem falhar, e os requerimentos de conectividade (r_{uv}, entre os terminais u e v) se referem apenas a eles. Aqui, os r_{uv} caminhos entre u e v tem que ser disjuntos nos elementos, isto é, não podem ter nem arestas nem vértices de Steiner em comum, mas podem ter terminais em comum. Assim, o problema consiste em achar um subgrafo de custo mínimo que contém pelo menos r_{uv} caminhos disjuntos nos elementos entre u e v, para todo u e v terminais.

4.1 O programa inteiro e a relaxação linear

Seja $G = (V, E)$ um grafo não orientado com custos não negativos c_e nas arestas. V é particionado em dois conjuntos disjuntos, R, os terminais, e S, os vértices de Steiner. Cada par de terminais tem um requerimento de conectividade r_{uv}, $u, v \in R$. Os elementos em $S \cup E$ são chamados de elementos. O problema consiste em encontrar um subgrafo H tal que, para cada par de terminais u e v, ainda há um caminho entre eles caso $r_{uv} - 1$ elementos sejam removidos; ou seja, há r_{uv} caminhos disjuntos nos elementos entre u e v.

Seja H uma solução viável. Suponha que um conjunto $A \subseteq S \cup E$ foi removido. O grafo $H - A$ deve ter pelo menos $r_{uv} - |A|$ caminhos disjuntos nos elementos para cada par u, v.

Seja $f(B) = max_{u \in B, v \notin B} r_{uv}$. O programa inteiro é o seguinte:

$$\begin{aligned}
&\min \textstyle\sum_{e \in E} c_e x_e \\
&\text{sujeito a } \textstyle\sum_{e \in \delta_{G-A}(B)} x_e \geq f(B) - |A|, \forall A \subseteq S \cup E, B \subseteq V - A, \\
&\qquad\qquad x_e \in \{0, 1\}, \forall e \in E.
\end{aligned}$$

A relaxação linear corresponde a trocar $x_e \in \{0, 1\}$ por $x_e \geq 0$.

O dual correspondente é o seguinte:

$$\begin{aligned}
&\max \textstyle\sum_{A \subseteq S \cup E, B \subseteq V - A} (f(B) - |A|) y_{BA} \\
&\text{sujeito a } \textstyle\sum_{BA: A \subseteq S \cup E, B \subseteq V - A, e \in \delta_{G-A}(B)} y_{BA} \leq c_e, \forall e \in E \\
&\qquad\qquad y_{BA} \geq 0, \forall A \subseteq S \cup E, B \subseteq V - A.
\end{aligned}$$

4.2 O algoritmo

O algoritmo é bem semelhante ao de Gabow e outros [9]. Também trabalha em fases. É definida a *deficiência* de uma restrição como a diferença entre seu lado esquerdo e seu lado direito. A deficiência de um conjunto $B \subseteq V$ é definida como a maior deficiência de uma restrição na qual B está envolvido. O algoritmo tem k fases. No início da fase p, a maior deficiência que pode existir é p, e no final da fase é $p - 1$. As fases são numeradas em ordem decrescente. Assim, no fim da fase 1, temos uma solução viável.

Seja H a solução parcial que temos no início da p-ésima fase. Seja h:

$$h(B) = \begin{cases} 1 & \text{se a deficiência de } B \text{ é } p \\ 0, & \text{caso contrário} \end{cases}$$

Em cada fase, tenta-se resolver o seguinte programa inteiro:

$$\begin{aligned}
&\min \textstyle\sum_{e\in E} c_e x_e \\
&\text{sujeito a } \textstyle\sum_{e\in\delta_{G-\varepsilon_H(B)}} x_e \geq h(B), \forall B \subseteq V, \\
&\qquad\qquad x_e \in \{0,1\}, \forall e \in E - H,
\end{aligned}$$

onde $\varepsilon_H(B)$ são os elementos vizinhos a B, isto é, vértices de Steiner vizinhos a algum vértice de B ou arestas com uma ponta em B e outra em $R - B$.

Da mesma forma que antes, obtemos o programa linear relaxando $x_e \in \{0,1\}$ para $x_e \geq 0$. O dual é o seguinte:

$$\begin{aligned}
&\max \textstyle\sum_{B\subseteq V} h(B) y_B \\
&\text{sujeito a } \textstyle\sum_{B:e\in\delta_{G-\varepsilon_H}(B)} y_B \leq c_e, \forall e \in E - H \\
&\qquad\qquad y_B \geq 0.
\end{aligned}$$

O algorimo para uma fase está descrito a seguir:

01 $I \leftarrow \emptyset$

02 Encontre todos os conjuntos ativos

03 Incremente seus y_B's até alguma aresta $e \in E - (\varepsilon_H(B) \cup I)$ fique justa

04 $I \leftarrow I \cup \{e\}$

05 Se há conjuntos violados com relação a I, vá ao passo 02

06 **Reverse delete**: Para arestas $e \in I$, em ordem reversa, faça: se $I - \{e\}$ não tem conjuntos não satisfeitos, então $I \leftarrow I - \{e\}$

As provas de corretude e razão de aproximação estão em [10].

5 Facetas de poliedros

Agora, veremos uma abordagem diferente para redes seguras, baseada em poliedros. Este é um trabalho de Grötschel, Monma e Stoer [1]. Aqui, não desenvolveremos toda a idéia, apenas qual é o rumo do trabalho.

Sâo dados um grafo $G = (V, E)$, com custos c_e para todo $e \in E$, e requerimentos de conectividade r_v, para todo $v \in V$. O problema consiste em encontrar uma rede $N = (V, F)$, $F \subseteq E$, tal que existem $r(s,t) = \min\{r_s, r_t\}$ caminhos disjuntos nas arestas (ou nos nós) entre s e t, para todo par de vértices distintos s e t, com o menor custo possível.

Dado $W \subseteq V$, o conjunto de arestas $\delta(W) = \{ij \in E | i \in W, j \in V - W\}$ é chamado **corte** (induzido por W).

Para $W, W' \subseteq V$ com $W \cap W' = \emptyset$ definimos $[W : W'] = \{ij \in E | i \in W, j \in W'\}$. Assim, $\delta(W) = [W : V - W]$.

Neste trabalho vamos considerar apenas as seguintes conectividades:

$$r_s \in \{0,1,2\} \text{ para todo } s \in V$$

Vamos definir o problema **2ECON** (respectivamente, **2NCON**) como sendo o problema de achar a rede N onde cada par de nós distintos s e t existem pelo menos $\min(r_s, r_t)$ caminhos disjuntos nas arestas (respectivamente, nos nós) entre s e t.

Extendemos a noção dos requerimentos r para conjuntos

$$
\begin{aligned}
r(W) &= \max\{r_s | s \in W\} \text{ para todo } W \subseteq V \\
con(W) &= \max\{r(s,t) | s \in W, t \in V - W\} \\
&= \min\{r(W), r(V-W)\} \text{ para todo } W \subseteq V, \emptyset \neq W \neq V.
\end{aligned}
$$

Vamos agora introduzir, para cada aresta $e \in E$, uma variável x_e e considerar o espaço vetorial $\mathbf{R}^E$. Todo subconjunto $F \subseteq E$ induz um **vetor incidente** $\chi^F = (\chi_e^F)_{e \in E} \in \mathbf{R}^E$ fazendo $\chi_e^F = 1$ se $e \in F$ e $\chi_e^F = 0$ caso contrário. Vice versa, cada vetor 0/1 $x \in \mathbf{R}^E$ indeuz um subconjunto $F^x = \{e \in E | x_e = 1\}$ de E. Para qualquer subconjunto $F \subseteq E$, definimos $x(F) = \sum_{e \in F} x_e$. Assim, o 2NCON pode ser formulado como o seguinte programa inteiro:

(IP) $\min \sum_{ij \in E} c_{ij} x_{ij}$
(i) $x(\delta(W)) \geq \text{con}(W)$ para todo $W \subseteq V$, $\emptyset \neq W \neq V$;
(ii) $x(\delta_{G-z}(W)) \geq 1$ para todo $z \in V$, e para todo $W \subseteq V - \{z\}$, $\emptyset \neq W \neq V - \{z\}$ com $r(W) = 2$ e $r(V - (W \cup \{z\})) = 2$;
(iii) $0 \leq x_{ij} \leq 1$ para todo $ij \in E$;
(iv) x_{ij} inteiro para todo $ij \in E$.

Segue do teorema de Menger que, para toda solução viável x de (IP), o subgrafo $N = (V, F^x)$ de G define uma rede satisfazendo 2NCON. Sem a restrição (ii) temos um programa inteiro para o 2ECON. Uma desiguladade do tipo (i) é chamada desigualdade de corte, uma do tipo (ii) é chamada desigualdade de nó-corte e uma do tipo (ii) é chamada desigualdade trivial.

Agora, vamos definir os seguintes politopos:
2NCON($G; r$)= conv$\{x \in \mathbf{R}^E | x$ satisfaz (i), (ii), (iii) e (iv) de (IP)$\}$
2ECON($G; r$)= conv$\{x \in \mathbf{R}^E | x$ satisfaz (i), (iii) e (iv) de (IP)$\}$

onde "conv" é a operação de fecho convexo. Dizemos que $F \subseteq E$ é viável para um desses politopos se χ^F é.

Agora vamos definir arestas "essenciais". Dizemos que $e \in E$ é essencial com respeito ao 2ECON($G; r$) se 2ECON($G - e; r$) $= \emptyset$. Da mesma forma, e é essencial com respeito ao 2NCON($G; r$) se 2NCON($G - e; r$) $= \emptyset$. Ou seja, e é essencial se sua remoção resulta em um grafo onde algum requerimento não pode ser satisfeito. Denotamos o conjunto E de arestas essenciais para 2ECON($G; r$) por 2EES($G; r$) e o conjunto de arestas essenciais para 2NCON($G; r$) por 2NES($G; r$). Seja $\dim(S)$ a dimensão de um conjunto $S \subseteq \mathbf{R}^n$. Em [6] é provado que o poliedro 2ECON($G; r$) é *full-dimensional* se e somente se 2EES($G; r$) é vazio, e também que 2NCON($G; r$) é *full-dimensional* se e somente se 2NES($G; r$) é vazio.

Seja $G = (V, E)$ um grafo e $W \subseteq V$ com $|W| \geq 2$, e seja $G' = (V, E')$, o grafo simples de G (G pode ter arestas múltiplas). Definimos:

$\lambda(G, W)$ = cardinalidade mínima de um subconjunto F de E, tal que dois nós de W estão desconectados em $G - F$;
$\kappa(G, W)$ = cardinalidade mínima de um conjunto $S \cup F$, onde $S \subseteq V$ e $F \subseteq E'$, tal que dois nós de W estão desconectados em $G' - (S \cup F)$.

Se $|W| < 2$, então definimos $\lambda(G, W)$ e $\kappa(G, W)$ como ∞. Essas funções são utilizadas especialmente em duas situações. Para facilitar, definimos:

$$\lambda_i(G) = \lambda(G, V_i) \text{ e } \kappa_i(G) = \kappa(G, V_i)$$

onde $V_i = \{v \in V | r_v \geq i\}$, $i = 0, 1, 2$.
Vamos assumir o seguinte:
(5.1) Sejam dados $G = (V, E)$ e $r \in \mathbf{Z}_+^V$.
(i) $r \in \{0, 1, 2\}^V$ e pelo menos dois nós diferentes s, t satisfazem $r_s = r_t = 2$;
(ii) se considerarmos o problema 2ECON, assumimos que G é 2-conexo e $\lambda_2(G) \geq 3$;
(iii) se considerarmos o problema 2NCON, assumimos que G é 2-conexo e $\kappa_2(G) \geq 3$.

5.1 Facetas

Uma desigualdade $a^T x \geq a$ é válida com relação ao poliedro P se $P \subseteq \{x | a^T x \geq a\}$; o conjunto $F_a = \{x \in P | a^T x \geq a\}$ é chamado a **face** de P definida por $a^T x \geq a$. Se $\dim(F_a)=\dim(P) - 1$ e $F_a \neq \emptyset$, então F_a é uma **faceta** de P, e $a^T x \geq a$ é chamada de definidor de faceta ou indutor de faceta.

Este teorema caracteriza as desigualdades trivias que definem facetas.

Teorema 4 *Seja (G, r) satisfazendo (5.1).*

1. *$x_e \leq 1$ define uma faceta de 2ECON($G; r$) e de 2NCON($G; r$) para todo $e \in E$.*
2. *$x_e \geq 0$ define uma faceta de 2ECON($G; r$) (respectivamente 2NCON($G; r$)) para $e \in E$ se e somente se para toda aresta $f \neq e$ o plitopo 2ECON($G - \{e, f\}; r$) (respectivamente 2NCON($G - \{e, f\}; r$)) é não vazio.*

O próximo teoremo caracteriza as desigualdades de corte que definem facetas.

Teorema 5 *Seja (G, r) satisfazendo (5.1) e $W \subseteq V$ com $\emptyset \neq W \neq V$.*

1. *Suponha con(W) = 2. Então $x(\delta(W)) \geq 2$ define uma faceta de 2ECON($G; r$) se e somente se*
   - *(a) $G[W]$ e $G[V - W]$ são conexos;*
   - *(b) $\lambda_1(G[W]) \geq 2$ e $\lambda_1(G[V - W]) \geq 2$;*
   - *(c) e é uma ponte de $G[W]$. Então f é uma ponte de $G[V - W]$, U, U' são os conjuntos de nós de duas componentes de $G[W] - e$, e Z, Z' são os conjuntos de nós de duas componentes de $G[V - W] - f$; e se $r(U) = r(Z) = 2$ (implicando $r(U') = r(Z') = 0$), então $|[U : Z]| \geq 1$.*
2. *Suponha con(W) = 1. Então $x(\delta(W)) \geq 1$ define uma faceta de 2ECON($G; r$) se e somente se*
   - *(a) $G[W]$ e $G[V - W]$ são conexos;*
   - *(b) $\lambda_1(G[W]) \geq 2$ e $\lambda_1(G[V - W]) \geq 2$;*
   - *(c) $\lambda_2(G[W]) \geq 3$ e $\lambda_2(G[V - W]) \geq 3$.*
3. *Suponha con(W) = 0. Então $x(\delta(W)) \geq 0$ não define uma faceta de 2ECON($G; r$) ou 2NCON($G; r$).*

4. *Suponha con(W) = 2. Então* $x(\delta(W)) \geq 2$ *define uma faceta de 2NCON(*$G; r$*) se e somente se*

   (a) *as condições 1(a), 1(b) e 1(c) estão satisfeitas;*

   (b) $\kappa_2(G[W]) \geq 2$ *e* $\kappa_2(G[V - W]) \geq 2$*;*

   (c) u *é um nó de articulação de* $G[W]$ *e* z *é um nó de articulação de* $G[V - W]$*, e se* U *e* Z *são conjuntos de nós componentes de* $G[W] - u$ *e* $G[V - W] - z$*, respectivamente, tais que* $r(U) = r(Z) = 2$*, então* $|[U : Z]| \geq 1$*, e (por causa de (b)) todos os outros componetes de* $G[W] - u$ *e* $G[V - W] - z$ *não contém nós* v *com* $r_v = 2$*;*

   (d) *nem para* $S = W$ *nem para* $S = V - W$ *a seguinte situação ocorre: Existem uma aresta* $e \in E(S)$ *e nós* $w_1, w_2 \in S$ *(não necessariamente distintos) e um nó* $w_3 \in V - S$ *tais que existe um componente* (S_1, E_1) *de* $(G[S] - e) - w_1$*, um componente* (S_2, E_2) *de* $(G[S] - e) - w_2$*, e um componente* (S_3, E_3) *de* $G[V - S] - w_3$ *com* $r(S_1) = r(S_2) = r(S_3) = 2$*,* $S_1 \cap S_2 = \emptyset$*, tais que em* $G - e$ *não há aresta entre* S_i *e* S_j*, para todo* $i \neq j, i, j \in \{1, 2, 3\}$*;*

   (e) *para* $S = WeS = V - W$ *o seguinte vale: se* $V - S$ *tem exatamente dois vizinhos em* S*, então um deles (*q*) é o único com* $r_q = 2$ *em* S*.*

5. *Suponha* $con(W) = r(W) = 1$*. Então* $x(\delta(W)) \geq 1$ *define uma faceta de 2NCON(*$G; r$*) se e somente se*

   (a) *as condições 2(a), 2(b) e 2(c) são satisfeitas;*

   (b) $\kappa_2(G[V - W] - e) \geq 2$ *para todo* $e \in E(V - W)$*.*

Finalmente, este teorema caracteriza quais desigualdades nó-corte definem facetas para 2NCON($G; r$).

Teorema 6 *Seja* (G, r) *satisfazendo (5.1) e seja um nó* $z \in V$*, e um conjunto* $W \subseteq V - \{z\}$ *com* $\emptyset \neq W \neq V - \{z\}$ *e* $r(W) = 2$*,* $r(V - (W \cup \{z\})) = 2$ *dados. Chamemos de* $V_i, i = 1, 2$*, o conjunto de nós* $v \in V$ *com* $r_v \geq i$*, e seja* $W' = V - (W \cup \{z\})$*.*

A desigualdade $x(\delta_{G-z}(W)) \geq 1$ *define uma faceta de 2NCON(*$G; r$*) se e somente se*

1. $G[W]$ *é conexo;*

2. $\lambda(G[W \cup \{z\}], V_1 \cup \{z\}) \geq 2$*;*

3. $\lambda_2(G[W]) \geq 2$*;*

4. $u \in W$ *é um nó de articulação de* $G[W \cup \{z\}]$ *separando dois nós de* $V_2 \cup \{z\}$*, e* $S \subseteq W$ *é o conjunto de nós de uma componente de* $G[W \cup \{z\}] - u$ *com* $r(S) = 2$*; então* $[W - S : W] = \emptyset$ *e* $r(W - (S \cup \{u\})) \leq 1$*;*

5. *a seguinte situação não ocorre: Existem uma aresta* $e \in E(W)$ *e dois nós* $w_1, w_2 \in W$ *(não necessariamente distintos), tais que, para* $i = 1, 2$*,* $(G[W \cup \{z\}] - e) - w_i$ *tem um componente com conjunto de nós* S_i*, onde* $S_i \subseteq W$ *e* $r(S_i) = 2$*; além disso,* $S_1 \cap S_2 = \emptyset$ *e* $e \in [S_1 : S_2]$*;*

6. condições 1 a 5 também valem para W' no lugar de W.

As provas desses teoremas, que são a base desse estudo, estão em [1].

6 Conclusão

Redes seguras são objetos importantes de estudo atualmente, por sua aplicação em redes telefônicas, redes de fibra ótica, e outras, onde há nós mais importantes que outros, e queremos que esses nós suportem um certo nível de falha.

Apesar de ser um problema difícil, para o qual não se conhece um algoritmo polinomial, existem vários algorimtos de aproximação interessantes, que conseguem boas aproximações. Este estudo será continuado, tentando cada vez mais um aprofundamento nos algoritmos e nas modelagens existentes.

Referências

[1] Grötschel, Martin; Monma, Clyde L.; Stoer, Mechthild Facet for polyhedra arising in the design of communication networks with low-connectivity constraints. SIAM J. Optim. 2 (1992), no. 3, 474–504.

[2] Goemans, Michel X.; Bertsimas, Dimitris J. Survivable networks, linear programming relaxations and the parsimonious property. Math. Programming 60 (1993), no. 2, Ser. A, 145–166. (Reviewer: Cheng En Yang) 90C35 (90C27)

[3] Gabow, Harold N.; Goemans, Michel X.; Williamson, David P. An efficient approximation algorithm for the survivable network design problem. Networks and matroids; Sequencing and scheduling. Math. Programming 82 (1998), no. 1-2, Ser. B, 13–40. 90C35 (68M10 68R10 90B10)

[4] Agrawal, A.; Klein, P.; Ravi, R. When tress collide: An approximation algorithm for the generalized Steiner problem in networks. Proc. 23rd ACM Symp. on Theory of Computing, 134-144 (1991)

[5] Berman, P.; Ramaiyer, V. Improved approximations for the Steiner tree problem. Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms, 325-334 (1992)

[6] Grötschel, M.; Monma, C. L.; Integer polyhedra asociated with certain network design problems with connectivity constraints. SIAM J. Discrete Math. 3, 502-523 (1990)

[7] Saran, H.; Vazirani, V.; Young, N. A primal-dual approach to approximation algorithms for network Steiner problems. Proc. of Indo-US workshop on Cooperative Research in Computer Science, Bangalore, India, 166-168 (1992)

[8] Aneja, Y.P. An integer linear programming approach to the Steiner problem in graphs. Networks 10, 134-144 (1980)

[9] Williamson, D. P.; Goemans, M. X.; Mihail, M.; Vazirani, V. V. A primal-dual approximation algorithm for generalized Steiner network problems. Combinatorica 15, 434-454 (1995)

[10] Jain, K.; Mandoiu, I.; Vazirani, V. V.; Williamson, D. P. A primal-dual schema based approximation algorithm for the element connectivity problem. Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms, 484-489 (1999)