O Problema de Steiner em Grafos Dirigidos

Alfredo Enrique Candia Vejar

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc.
(presidente)

Cláudio Bornstein, Dr. Rer. Nat.

Prof. Antônio de Oliveira, D.Sc.

Prof. Jayme Szwarcfiter, Ph.D.

Prof. Luis Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 1991

CANDIA VEJAR, ALFREDO ENRIQUE

O Problema de Steiner em um Grafo Dirigido

IV, 142 pgs., 29.7 cm, (COPPE/UFRJ, D. Sc., ENGENHARIA DE SISTEMAS E COMPUTAÇÃO, 1991)

TESE – Universidade Federal do Rio de Janeiro, COPPE

1 – Otimização em Grafos 2 – Problema de Steiner 3 – Testes de Redução

I. COPPE/UFRJ II. Título(Série).

Aos meus pais, irmãs e Fafi

Agradecimentos

Ao Profesor Nelson Maculan pelo apoio, confiança e por iniciar-me no campo da pesquisa.

A Paulo Mello de Souza pela sua contribuição na implementação dos algoritmos, escritura da tese, comentários e também pela sua amizade.

Ao amigo Hugo Bravo pelas discussões sobre métodos de redução.

Aos membros da Banca de Tese pela sua importante colaboração.

'A CAPES pelo seu apoio financeiro durante uma parte importante de meus estudos.

Ao Departamento de Matemática da Universidade de Tarapacá pelas facilidades dadas para a conclusão da tese.

Finalmente, eu gostaria de expressar meu reconhecimento ao Programa de Engenharia de Sistemas e Computação pelo bom nivel de Ensino e Pesquisa.

Resumo da Tese apresentada à COPPE como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D. Sc.)

O Problema de Steiner em Grafos Direcionados

Alfredo Enrique Candia Vejar

Março de 1991

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

Estudaremos um problema de Otimização Combinatória da área de *Desenho Ótimo de Redes* : O Problema de Steiner em Grafos Direcionado.

Dado um grafo direcionado $D = (V, E)$ com raiz $r \in V$ ponderado nos arcos com função de pesos w, o Problema de Steiner em Grafos Direcionados consiste em encontrar caminhos direcionados da raiz r até os vértices do conjunto $V_o \subseteq V$, de peso total mínimo.

Primeiro, revisamos alguns elementos da Teoria dos Grafos Direcionados, definimos formalmente o problema e calculamos sua complexidade computacional.

A seguir, revisamos diversas formulações e algoritmos para o problema e comentamos experiências computacionais.

Depois investigamos testes de redução para o problema os quais permitem diminuir o tamanho do grafo direcionado através da eliminação de vértices e

arcos. Exibimos dois tipos de testes. Uns são uma extensão direta de testes já conhecidos para o caso particular de grafos não-direcionados. Outros são especialmente desenvolvidos para o caso direcionado.

Finalmente, apresentamos um algoritmo exato para o problema. Tal algoritmo soluciona uma formulação de arborescência geradora restrita para o problema. Para instâncias geradas aleatoriamente aplicamos os testes de redução desenvolvidos e depois aplicamos o algoritmo exato para encontrar a solução ótima. A formulação também permite encontrar soluções quase viáveis.

Abstract of Thesis presented to COPPE as partial fulfilment of the requirements for the degree of Doctor of Science (D. Sc.)

The Steiner Problem in Directed Graphs

Alfredo Enrique Candia Vejar

March, 1991

Thesis Supervisor: Nelson Maculan Filho

Department: Programa de Engenharia de Sistemas e Computação

We study a problem of Combinatorial Optimization related to Optimal Network Design: The Steiner Problem in Directed Graphs.

Given a Directed Graph $D = (V, E)$ with root $r \in V$, arc-weighted with weight function w, The Steiner Problem in Directed Graphs consists in finding paths directed from r to all the vertices of the set $V_0 \subseteq V$, of minimum total weight.

First we review some elements of the theory of Directed Graphs; we define the problem and discuss its computational complexity.

Then we study different formulations and algorithms for the problem and discuss computational experiences.

We investigate reduction tests for the problem which enable us to reduce the size of the instances by eliminating vertices and arcs. We present two classes of tests. First class corresponds to a direct extension of known tests for the special case of graphs. The second class corresponds to a tests specially designed

for the directed case.

Finally, we propose an exact algorithm for the problem. The algorithm solves a restricted spanning arborescence formulation of the problem. For randomly generated instances, we apply the reduction tests developed and then we apply the exact algorithm for finding the optimal solution. The formulation also gives us near-feasible solutions.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo I

Introdução

I.1 Elementos da Teoria dos Dígrafos

Resumiremos nesta seção alguns conceitos básicos da teoria dos grafos direcionados. Abreviaremos as palavras grafo direcionado como sendo a palavra **dígrafo**. Este tipo de grafos será nosso objeto de estudo durante o transcurso deste trabalho. Conceitos de dígrafos podem ser encontrados em Gondran e Minoux [22], Even [17] e Szwarcfiter [36] entre outros. Um tratamento particular de problemas de arborescências em dígrafos pode ser encontrado em Candia [8].

Um dígrafo $D = (V, E)$ consiste em um conjunto finito e não vazio de elementos chamados **vértices** ou **nós**, e de um conjunto finito de elementos de $V \times V$, chamados **arcos**. Cada arco $e \in E$ é um par ordenado (v, w), onde v e w são vértices de V. Dizemos que um arco $e = (v, w)$ **sai** de v e **entra** em w. O vértice v é o **vértice inicial** de e e w é o **vértice final** de e ; v e w se dizem **adjacentes**. O **grau interno** de um vértice v é a quantidade de arcos que entram em v e se denota $d^-(v)$. O **grau externo** de v é o número de arcos que saem de v e se denota $d^+(v)$. Dois arcos distintos $e_1 = (v_1, w_1)$ e $e_2 = (v_2, w_2)$ são **paralelos** se $v_1 = v_2$ e $w_1 = w_2$. Um arco e é um **laço** se e sai e entra num mesmo vértice, ou seja, $e = (u, u)$. Se $d^-(v) = d^+(v) = 0$ então v se chama **vértice isolado**. Se $d^-(v) = 1$ e $d^+(v) = 0$ então v se chama **folha**.

Um dígrafo pode ser representado graficamente por pontos representando os vértices e por setas representando os arcos. Por exemplo, consideremos

o dígrafo definido por $D = (V, E), V = \{v, w, z, y\}$, $E = \{a, b, c, d\}, a = (v, w)$, $b = (w, v), c = (y, v)$, $d = (w, y)$.

Uma representação válida para D é dada na figura I.1

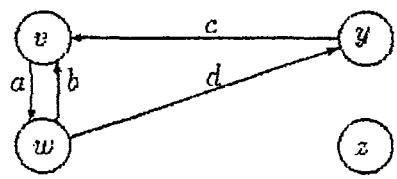


Figura I.1: Um exemplo de dígrafo.

Observamos que z é um vértice isolado.

Um dígrafo é **simples** se ele não contém laços nem arcos paralelos. Trabalharemos normalmente com dígrafos simples. Um **subdígrafo** $D' = (V', E')$ do dígrafo $D = (V, E)$ é um dígrafo tal que $V' \subseteq V$ e $E' \subseteq E$.

Dado $W \subseteq V$, $D - W$ denota o dígrafo obtido de D eliminando os vértices de W e os arcos (u, v) que tem $u \in W$ ou $v \in W$. Se em particular $W = \{w\}$ então escreveremos $D - w$ em vez de $D - \{w\}$. Analogamente, para $A \subseteq E$, $D - A$ denota o dígrafo obtido de D eliminando os arcos de A. Escreveremos $D - e$ para $D - \{e\}$. Também, se $u, v \in V$ e $e = (u, v) \notin E$ então denotamos por $D + e$ o dígrafo $D' = (V', E')$ tal que $V' = V$ e $E' = E \cup \{e\}$. $D + W$ denota o dígrafo obtido de D adicionando os vértices de W. Se em particular $W = \{w\}$ então escreveremos $D + w$ em vez de $D \cup \{w\}$. Seja $A \subseteq E$, $D[A]$ denotará o subdígrafo de D que tem como arcos A e conjunto de vértices os vértices de V que são vértices iniciais ou terminais dos arcos de A.

Um dígrafo $D = (V, E)$ é **completo** se para todo par de vértices $u, v \in V$, $u \neq v$, $e = (u, v) \in E$.

Um **caminho** em um dígrafo $D = (V, E)$ é uma sequência $C = (v_0, e_1, v_1, e_2, \ldots, e_k, v_k)$, $k \geq 1$, de vértices e arcos todos eles distintos. Se $v_0 = v_k$ e $v_0, \ldots, v_{k-1}$ são todos distintos então C se chama **circuito**. O **comprimento** de C em cada um desses casos é o número de arcos na sequência. Dizemos que o vértice u

é um **predecessor** do vértice v em D, e que v é um **sucessor** de u em D, se existe um caminho de u a v em D.

Um dígrafo $D = (V, E)$ é **fortemente conexo** se para cada par de vértices $u, v \in V$, existe um caminho de u até v. Um dígrafo D é **conexo** se o grafo não direcionado obtido de D ao eliminar as direções é conexo. Um vértice v em D é um vértice **raiz** se existir em D caminhos de r até cada outro vértice de D. Uma **r-árvore dirigida** ou **r-arborescência** em D é um dígrafo com raiz r, sem circuitos e onde cada vértice distinto da raiz tem grau interno 1. Se $D = (V, E)$ é um dígrafo então $D' = (V', E')$ é um **subdígrafo gerador** de D se $V' = V$ e $E' \subseteq E$. Assim, uma **r-arborescência geradora** de um dígrafo $D = (V, E)$ é um subdígrafo $T = (V, E')$ de D tal que T é uma r-arborescência e contém todos os vértices de D. Observamos que se T é uma r-arborescência geradora de D então $| E' | = | V | - 1$.

Exemplo. A figura I.2 ilustra uma r-arborescência geradora.

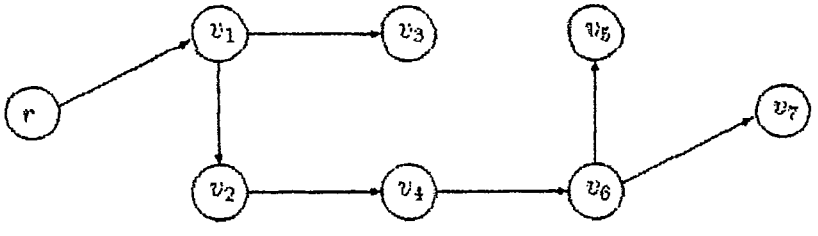


Figura I.2: Um exemplo de arborescência geradora.

Um **dígrafo ponderado** é um dígrafo no qual os vértices e/ou arcos tem uma função real associada a eles. Nós trabalharemos com dígrafos ponderados nos arcos; assim, se $D = (V, E)$ é um dígrafo então existe uma função w real tal que cada arco $e \in E$ tem associado um número w_e. Se $e = (u, v)$ então é possível escrever w_{uv}. Costuma-se denotar um dígrafo ponderado na forma $D = (V, E, w)$; as vezes se chama D de **rede**. Se $D' = (V', E', w)$ é um subdígrafo de D então o **peso** (ou **custo** ou **comprimento**) de D' está dado por

$$w(D) = \sum_{e \in E'} w_e.$$

I.2 Definição do Problema de Steiner em Dígrafos

O Problema de Steiner em Dígrafos (PSD) é uma generalização do **Problema de Steiner em Grafos (PSG)**.

No PSG são dados um grafo conexo $G = (V, E)$, uma função de pesos $w : E \longrightarrow R_+$ e um conjunto $V_0 \subseteq V$ de vértices chamados **vértices demanda**. Deseja-se determinar um conjunto $A \subseteq E$ tal que:

- $G' = (V[A], A)$ contém um caminho entre cada par de vértices de V_0
- $\sum_{e \in A} w_e$ é mínima.

É conhecido que a solução do PSG é uma árvore que gera V_0 (podendo conter vértices de $V - V_0$). Assim, se fala do PSG como o **Problema da Árvore de Steiner**.

O PSG tem sido amplamente estudado, ver os trabalhos de Ferreira [18], Maculan [30] e Winter [40] para histórico, algoritmos e aplicações do problema. Faremos aqui apenas comentários que relacionam o PSG a dois problemas que tem muitas aplicações. O PSG pode ser generalizado naturalmente como segue:

> Dado um conjunto de pontos (cidades, por exemplo) $A_1, A_2, \ldots, A_n$, deseja-se construir um sistema de segmentos de reta (rede de estradas) de comprimento total mínimo de tal forma que qualquer ponto esteja conectado a qualquer outro através deste sistema. Veja a figura I.3 para um exemplo.

Este problema é conhecido como o Problema de Steiner no Plano Euclidiano (PSP), ver Winter [39] para detalhes.

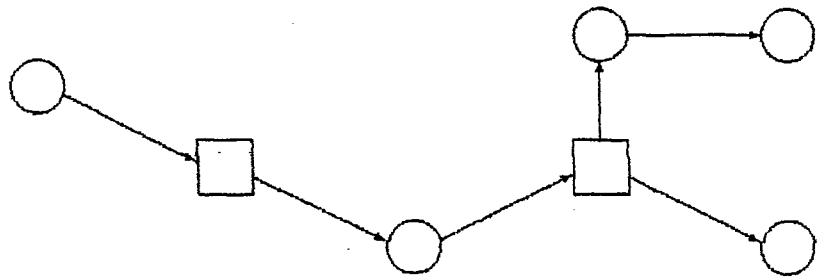


Figura I.3: O problema de Steiner no plano euclidiano.

O PSP tem uma variante na qual a métrica euclidiana é sustituída pela métrica retilínea. Dados dois pontos (x_1, y_1) e (x_2, y_2), a distância retilínea d_r entre estes pontos é definida por: $d_r = \mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid$. Tal variante foi sugerida em 1966 por Hanan [25] e leva o nome de **Problema de Steiner Retilíneo**, ver também De Souza [35] para novas heurísticas. A figura I.4 mostra um exemplo

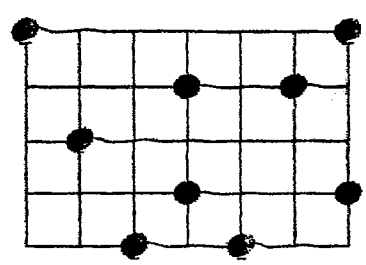


Figura I.4: O problema de Steiner retilíneo.

O PSD aparece naturalmente a partir do PSG quando os pesos nos arcos não são simétricos, ou seja, $w_{uv} \neq w_{vu}$. Analisemos detalhadamente as condições nas quais se define o PSD. Consideremos a rede $D = (V, E, w)$, onde w é uma função de pesos positiva, ou seja, cada arco $e = (u, v)$ tem associado um peso $w_e > 0$. O conjunto V é particionado como $V = \{r\} \cup V_0 \cup V_f$, onde r é uma raiz do dígrafo D. O PSD pode ser formulado como segue:

Dado o dígrafo ponderado $D = (V, E, w)$ com raiz r, deseja-se encontrar caminhos de r até cada vértice de V_0 de maneira que o peso total dos arcos pertencentes a esses caminhos seja mínimo.

Observamos que o PSD sempre tem solução devido a nossa suposição que r é raiz de D. Os vértices de V_f representam pontos alternativos para unir r com os vértices de V_0 e por isso eles são chamados de **vértices facultativos.**

Denotemos por D_0 o subdígrafo de D definido por $D_0 = (V_0 + r, E[V_0 + r])$. É importante notar que a não exigência de ter uma solução viável para o problema, fornecida por D_0, poderia implicar a participação a priori de elementos de V_f na solução ótima. Consideremos o seguinte exemplo dado na figura I.5.

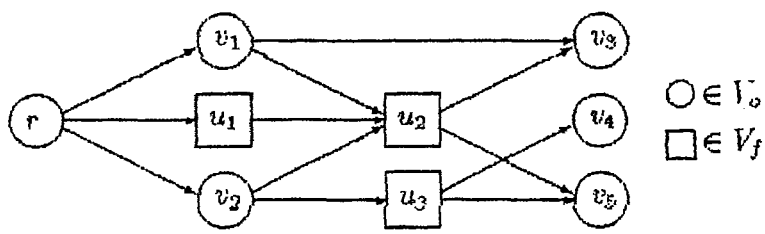


Figura I.5: Uma instância para o PSD.

É evidente que u_2 ou u_3 (ou ambos) devem participar da solução ótima já que D_0 não possui r-arborescências geradoras.

No entanto, qualquer dígrafo ponderado $D = (V, E, w)$ que não tenha raiz para D_0 pode ser simplesmente modificado adicionando arcos de r aos vértices não alcançados em D_0. Estes arcos terão pesos suficientemente grandes (por exemplo, $\sum_{e \in E} w(e)$). Assim, a partir do dígrafo $D = (V, E, w)$ sempre é possível construir uma instância do PSD onde D_0 tenha raiz.

O PSD tem casos particulares importantes. Se $\mid V_0 \mid = 1$ então o PSD se reduz ao problema de encontrar um caminho de peso mínimo entre dois vértices, os vértices r e $v \in V_0$, para o qual se sabe que existem algoritmos polinomiais, ver Dijkstra [12]. Se $V_f = \emptyset$ então chegamos ao problema da **r-arborescência geradora de peso mínimo**, ver por exemplo Edmonds [16].

Outro caso particular importante do PSD é o já definido **Problema de Steiner em Grafos** (PSG). Seja $G = (V, E, w)$ uma instância arbitrária do PSG.

A partir de G construímos um dígrafo ponderado $D = (V', E', w')$ na forma seguinte. Primeiro, definimos $V' = V$, depois $E' = \{(u, v) \mid [u, v] \in E\} \cup \{(v, u) \mid [u, v] \in E\}$ e $w'(u, v) = w(u, v)$ e $w'(v, u) = w(u, v)$; r é definido como sendo qualquer vértice de V_0. Então a solução do PSD para D dá a solução do PSG para $G = (V, E, w)$. Para isso, é suficiente remover as direções dos arcos em D. As arestas resultantes definem a solução para G.

Antes de terminar esta seção daremos um exemplo de uma instância do PSD, ilustrando sua solução.

Exemplo. Uma instância do PSD e sua solução ótima T aparecem na figura I.6.

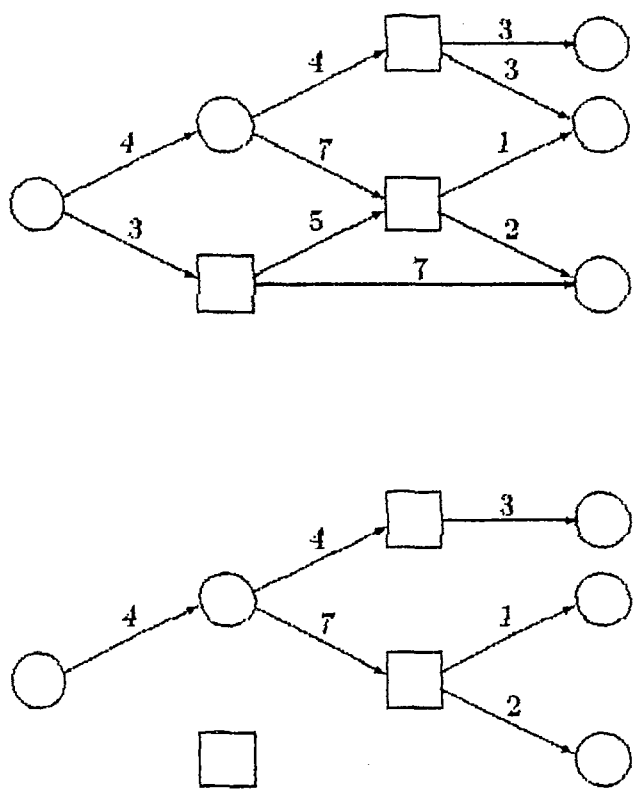


Figura I.6: Exemplo de solução ótima.

O peso da solução ótima é $w(T) = \sum_{e \in E[T]} w(e) = 21$.

Observamos que na solução ótima participam dois vértices de V_f. Estes vértices recebem o nome de **ponto de Steiner**. Ocupando esta definição podemos afirmar que solucionar o PSD é equivalente a encontrar os pontos de Steiner,

ou seja, ser capaz de decidir quais vértices de V_f participarão da solução ótima. Denotemos por V_f^* o conjunto de tais vértices, então a solução ótima do PSD é a r-arborescência geradora mínima do subdígrafo de D gerado por $\{r\} \cup V_0 \cup V_f^*$, e será chamada **$r$-arborescência de Steiner**.

I.3 Complexidade Computacional do Problema

Acabamos de mostrar que o PSD tem como caso particular o PSG. Logo, é suficiente provar que o PSG é $\mathcal{NP}$-Completo para ter que o PSD também é $\mathcal{NP}$-Completo.

Veremos que o P_dSG, o **problema de decisão** associado ao PSG é $\mathcal{NP}$-Completo. Apresentaremos uma redução do **Problema da Cobertura Exata** ao PSG seguindo a demonstração dada por Karp [27] e Ferreira [18].

Denotaremos por PCE o problema de decisão associado ao problema da Cobertura Exata. Definamos formalmente o P_dSG e o PCE.

P_dSG: Dado um grafo ponderado $G = (V, E, w)$, um subconjunto $Z \subseteq V$ e $k \in R_+$, encontrar (se existe) um subconjunto $A \subseteq E$ tal que o subgrafo $G[A]$ contém um caminho entre cada par de vértices de Z e $\sum_{e \in A} w_e \leq k$.

PCE: Dado um conjunto $U = \{u_1, u_2, \ldots, u_t\}$ e uma família $F = \{V_1, V_2, \ldots, V_r\}$ de subconjuntos de U, encontrar (se existe) uma subfamília de F que seja uma partição de U.

Mostremos um exemplo do PCE.

Exemplo. Consideremos $U = \{a, b, c, d, e, f, g\}$ e $F = \{\{a, b, c, d\}, \{b, c, d\}, \{e, f\}, \{a, g\}, \{b, e\}, \{c, d\}, \{g\}\}$, então a subfamília $\{\{b, c, d\}, \{e, f\}, \{a, g\}\}$ é uma cobertura exata de U.

Teorema I.1. *P_dSG é $\mathcal{NP}$-Completo.*

Prova. Primeiro vejamos que $P_dSG \in \mathcal{NP}$. Dado $A \subseteq E$, é possível verificar em tempo polinomial $O(|\ E\ |)$ se $G[A]$ contém um caminho entre cada par de vértices

de Z através da utilização de um algoritmo de busca. Também se verifica em tempo $O(|\ E\ |)$ se $\sum_{e \in A} w_e \leq k$. Assim, $P_dSG \in \mathcal{NP}$.

Vejamos agora que PCE pode ser reduzido ao P_dSG em tempo polinomial. Dada uma instância $\mathcal{I}_c$ do PCE, $U = \{u_1, u_2, \ldots, u_t\}$ e $F = \{V_1, V_2, \ldots, V_r\}$ construímos a seguinte instância $\mathcal{I}_s$ do P_dSG :

- Um grafo G com:

  $V = \{u_1, u_2, \ldots, u_t\} \cup \{n_0\} \cup \{V_1, V_2, \ldots, V_r\}$

  $E = \{[n_0, V_i] \mid i = 1, \ldots, r\} \cup \{[u_i, V_j] \mid u_i \in V_j, i = 1, 2, \ldots, t; j = 1, \ldots, r\}$.

- Uma função de pesos w definida por:

  $$w(e) = \begin{cases} |\ V_i\ | & \text{se } e \text{ é do tipo } [n_0, V_i] \text{ para algum } i = 1, \ldots, r \\ t & \text{caso contrário} \end{cases}$$

- $Z = \{u_1, u_2, \ldots, u_t, n_0\}$

- $k = t^2 + t$

Verifiquemos que $\mathcal{I}_c$ tem solução se e somente se $\mathcal{I}_s$ tem solução.

($\Rightarrow$) Seja $\Gamma = \{V_{i_1}, V_{i_2}, \ldots, V_{i_t}\}$ uma subfamília de F a qual é cobertura exata de U. Mostremos que o subgrafo $G[I(\Gamma)]$ é solução de $\mathcal{I}_s$, onde

$$I(\Gamma) = \{e \in E \mid e \text{ incide em algum vértice } V_{ij} \text{ que pertence a } \Gamma\}.$$

Com efeito, como Γ é cobertura exata de U, existe em $G[I(\Gamma)]$ exatamente uma aresta incidente a cada vértice correpondente a um elemento de U (tais arestas tem custo total t^2). Como cada vértice correspondente a um elemento de Γ está conectado a n_0, existe um caminho entre n_0 e cada vértice correspondente a um elemento de U, e portanto $G[I(\Gamma)]$ contém um caminho entre cada par de vértices de Z. Como Γ é cobertura exata de U, temos que o peso total das arestas incidentes a n_0 em $G[I(\Gamma)]$ é t. Logo, $\sum_{e \in E[I(\Gamma)]} w_e = t^2 + t$ e $G[I(\Gamma)]$ é solução de $\mathcal{I}_s$.

($\Leftarrow$) Agora, seja T uma solução de $\mathcal{I}_s$. Tomemos $\Gamma = \{V_j \mid [n_0, V_j] \in E[T]\}$. Como n_0 e os vértices correspondentes a elementos de U são vértices obrigatórios, existe

em T um caminho entre n_0 e cada um desses vértices. Devido á estrutura do grafo G temos que Γ é uma cobertura de U e assim $\sum_{V_j \in \Gamma} \mid V_j \mid \geq t$. Por outro lado, notamos que em cada vértice correspondente a um elemento de U de incidir pelo menos uma aresta de T. Como tais arestas tem custo t e $\mid U \mid = t$, o custo total destas arestas é pelo menos t^2. Como $\sum_{e \in E[T]} w_e \leq t^2 + t$, não podemos ter em T mais que t arestas incidentes aos vértices correspondentes aos elementos de U, dado que n_0 é obrigatório. Com isso, o custo total dessas arestas é exatamente t^2.

Como $\sum_{e \in E[T]} w(e) \leq t^2 + t$, então $\sum_{e \in E[T] \cap I(n_0)} w(e) \leq t$. Logo, $t \geq \sum_{e \in E[T] \cap I(n_0)} w(e) = \sum_{V_j \in \Gamma} \mid V_j \mid \geq t$. Portanto, $\sum_{V_j \in \Gamma} \mid V_j \mid = t$ e Γ é cobertura exata de U, e é uma solução de $\mathcal{I}_c$ ∎

Capítulo II

Formulações e Algoritmos para o Problema

II.1 Formulações para o Problema

As formulações para o PSD são na sua grande maioria formulações de Programação Linear Inteira (PLI). Analisaremos três formulações que aparecem no trabalho de Maculan [30]. Primeiro daremos outras definições particulares de dígrafos. Seja $D = (V, E)$ um dígrafo com raiz r. Seja $I(i)$ o conjunto de todos os vértices $j \in V$ tais que $(j, i) \in E$ e $O(i)$ o conjunto de todos os vértices $j \in V$ tais que $(i, j) \in E$. Denotemos por $C = (X, \overline{X}) \subseteq E$ o **conjunto corte** em D tal que $X \cup \overline{X} = V$, $X \cap \overline{X} = \emptyset$, $r \in X$ e $V_0 \cap \overline{X} \neq \emptyset$, onde $(i, j) \in C$ se e somente se $i \in X$ e $j \in \overline{X}$. Observamos que existem $O(2^{n-1})$ cortes possíveis, onde $n = \mid V \mid$.

Faremos $y_{ij} = 1$ se o arco $(i, j) \in E$ está na solução ótima e $y_{ij} = 0$ caso contrário. Para todas as formulações consideraremos $I(r) = \emptyset$ já que toda solução ótima para o PSD não vai ter arcos entrando na raiz r.

Formulação P_1

Esta formulação foi primeiro apresentada por Nguyen, Arpin e Maculan [32] para o PSD. Consideramos um **problema de fluxos em redes** no qual a raiz r oferece $\mid V_0 \mid$ unidades de fluxo e cada vértice $v \in V_0$ demanda uma unidade de fluxo.

(P_1) minimizar $\sum_{(i,j)\in E} w_{ij}\ y_{ij}$

sujeito a

$$\begin{aligned}
\sum_{j\in\mathcal{O}(i)} x_{ij} - \sum_{j\in\mathcal{I}(i)} x_{ji} &= 0 \quad i \in V \setminus V_0 \\
\sum_{j\in\mathcal{O}(k)} x_{kj} - \sum_{j\in\mathcal{I}(k)} x_{jk} &= -1 \quad k \in V_o \\
0 \le x_{ij} \le |V_o|\, y_{ij} & \quad (i,j) \in E \\
y_{ij} \in \{0,1\} & \quad (i,j) \in E
\end{aligned}$$

Formulação P_2

Esta formulação para o PSD foi independente apresentada por Claus e Maculan [11] e Wong [42]. Consideremos um problema de síntese de redes com demanda de fluxo de um único bem não simultâneo. Seja x_{ij}^k a quantidade do bem k entre r e $k \in V_0$ sobre o arco (i,j). A raiz r oferece uma unidade de cada bem $k \in V_0$.

(P_2) minimizar $\sum_{(i,j)\in E} w_{ij}\ y_{ij}$

sujeito a

$$\begin{aligned}
\sum_{j\in\mathcal{O}(i)} x_{ij}^k - \sum_{j\in\mathcal{I}(i)} x_{ji}^k &= 0 \quad , i \in V - \{r,k\}, k \in V_0 \\
\sum_{j\in\mathcal{O}(k)} x_{kj}^k - \sum_{j\in\mathcal{I}(k)} x_{jk}^k &= -1 \quad k \in V_o \\
0 \le x_{ij}^k \le |V_o|\, y_{ij} & \quad (i,j) \in E, k \in V_0 \\
y_{ij} \in \{0,1\} & \quad (i,j) \in E
\end{aligned}$$

Formulação P_3

Esta formulação foi apresentada para o PSD por Nguyen, Arpin e Maculan [32] e Wong [42].

(P_3) minimizar $\sum_{(i,j)\in E} w_{ij}\ y_{ij}$

sujeito a

$$\begin{aligned}
\sum_{(i,j)\in C_q} y_{ij} &\ge 1 \quad , q = 1,2,\ldots,p \\
y_{ij} \in \{0,1\} & \quad (i,j) \in E
\end{aligned}$$

Observamos que se substituirmos $y_{ij} \in \{0,1\}$ por $0 \le y_{ij} \le 1$ em P_1, P_2, P_3 teremos três relaxações de Programação Linear para o PSD que denotaremos

por LP_1, LP_2, LP_3 respectivamente. Seja $v(\cdot)$ o valor da solução ótima para o problema $(\cdot)$. A seguinte proposição dá limites inferiores para o valor da solução ótima do PSD.

Proposição II.1. $v(LP_1) \leq v(LP_2) = v(LP_3)$.

Prova. Vejamos primeiro que $v(LP_1) \leq v(LP_2)$. A partir de LP_2 temos

$$\begin{aligned}
\sum_{k\in V_0}\Big(\sum_{j\in O(i)} x_{ij}^k - \sum_{j\in I(i)} x_{ji}^k\Big) &= 0 \qquad i \in V_f \\
\sum_{k\in V_0}\Big(\sum_{j\in O(i)} x_{ij}^k - \sum_{j\in I(i)} x_{ji}^k\Big) &= -1 \qquad i \in V_0 \\
0 \leq \sum_{k\in V_0} x_{ij}^k \leq |V_0| \cdot y_{ij} &\qquad (i,j) \in E
\end{aligned}$$

Logo temos

$$\begin{aligned}
\sum_{j\in O(i)}\Big(\sum_{k\in V_0} x_{ij}^k\Big) - \sum_{j\in I(i)}\Big(\sum_{k\in V_0} x_{ji}^k\Big) &= 0 \quad , i \in V_f \\
\sum_{j\in O(i)}\Big(\sum_{k\in V_0} x_{ij}^k\Big) - \sum_{j\in I(i)}\Big(\sum_{k\in V_0} x_{ji}^k\Big) &= -1 \quad , i \in V_0
\end{aligned}$$

Se definimos $x_{ij} = \sum_{k\in V_0} x_{ij}^k$ (e portanto $x_{ji} = \sum_{k\in V_0} x_{ji}^k$) então LP_2 é um caso particular do LP_1 e assim $v(LP_1) \leq v(LP_2)$.

Mostremos agora que $v(LP_2) = v(LP_3)$. Primeiro verifiquemos que $v(LP_2) \leq v(LP3)$. Seja $\overline{y}_{ij}$ uma solução viável de LP_3, ou seja,

$$\sum_{(i,j)\in C_q} \overline{y}_{ij} \geq 1, q = 1, \ldots, p \quad \text{e}$$

$$0 \leq \overline{y}_{ij} \leq 1, \quad (i,j) \in E.$$

Se consideramos que $\overline{y}_{ij}$ representa a capacidade do arco (i,j) em LP_2 então deve ser possível enviar uma unidade do fluxo do bem k de r a $k \in V_0$. Aplicando o teorema do fluxo máximo–corte mínimo, ver Ford e Fulkerson [20], a capacidade total de qualquer corte separando os nós r e k deve ser no mínimo 1; $\overline{y}_{ij}$ será uma solução viável de LP_2 e logo $v(LP_2) \leq v(LP_3)$. Verifiquemos agora que $v(LP_2) \geq v(LP_3)$. Seja $(x_{ij}^k, \overline{y}_{ij})$ uma solução viável de LP_2, então para todo $k \in V_0$ o problema de fluxo associado dá $\sum_{(i,j)\in C_q} \overline{y}_{ij} \leq 1$, $q = 1, \ldots, p$. $\overline{y}_{ij}$ deve ser uma solução viável de LP_3 e logo $v(LP_2) \geq v(LP_3)$. ∎

É claro que se $v(P)$ é o valor ótimo para o PSD, então $v(P) = v(P_1) = v(P_2) = v(P_3)$. Da proposição 1 anterior concluímos que:

$$v(LP_1) \leq v(LP_2) = v(LP_3) \leq v(P),$$

ou seja, $v(LP_2)$ e $v(LP_3)$ são melhores limites inferiores para $v(P)$ que $v(LP_1)$. Diferentes algoritmos são aplicados na resolução das três formulações anteriores. Guyard [23] e Nguyen, Arpin e Maculan [32] aplicaram o método de Decomposição de Benders [6] para a formulação P_2 (possível de aplicar também a P_1). Dror, Gavish e Choquette [10] fizeram experimentos com relaxações Lagrangeanas associadas às formulações anteriores do PSD. Eles obtiveram resultados numéricos satisfatórios para dígrafos de até 100 vértices e 400 arcos. Wong [42] dá um algoritmo ascendente dual para o PSD a partir da formulação P_2. Vejamos a seguir o esquema do algoritmo de Wong.

II.2 Algoritmo de Wong

Consideremos o dual de LP_2 :

(DLP_2) maximizar $\sum u_i^k$

sujeito a

$$\begin{array}{rcll} u_j^k - u_i^k - v_{ij}^k & \leq & 0 & (i,j) \in E, i \neq r, k \in V_0 \\ u_j^k - v_{rj}^k & \leq & 0 & j \in O(r), \quad k \in V_0 \\ \sum\limits_{k \in V_0} v_{ij}^k & \leq & w_{ij} & (i,j) \in E \\ v_{ij}^k & \leq & 0 & (i,j) \in E, \ k \in V_0 \end{array}$$

onde a variável dual u_i^k está associada com a equação de conservação do fluxo para o bem k no vértice i.

Se define um dígrafo auxiliar $D' = (V, E')$, com $E' \subseteq E$. Se o vértice i é um predecessor do vértice j mas não acontece o contrário então se diz que o vértice i '**pendura**' do nó j. Além disso, uma componente fortemente conexa T é também uma componente enraizada se T contém um elemento de V_0 mas nenhum

elemento de $V_0 + r$ 'pendura' de um elemento de T. Seja $C(k)$ o conjunto de vértices predecessores do vértice k. **O conjunto corte de k**, $CS(k)$, é um conjunto de arcos (i, j) que satisfaz:

$$(i, j) \in E, (i, j) \notin E', j \in C(k) \text{ e } i \in \overline{C}(k).$$

O procedimento ascendente de Wong para calcular uma solução aproximada de (DLP_2) é o seguinte:

Passo 0 (Início).

$$\begin{array}{rcll} u_i^k & := & 0 & k \in V_0,\ i \in V \\ v_{ij}^k & := & 0 & k \in V_0,\ (i,j) \in E \\ R(i,j) & := & w_{ij} - \sum\limits_{k \in V_0} v_{ij}^k & (i,j) \in E \end{array}$$

Formamos o dígrafo auxiliar $D = (V, E')$ com $E' = \emptyset$ (no início todos os vértices são componentes fortementes conexas e todos os elementos de V_0 são componentes enraizadas).

Passo 1.

Escolhemos uma componente enraizada T. Se não existem componentes enraizadas então paramos.

Passo 2.

Escolhemos um vértice $k \in V_0 \cap T$. Seja

$$R(i^*, j^*) = min\{R(i, j) \mid (i, j) \in CS(k)\};$$

Para cada nó $h \in C(k)$, $u_h^k := u_h^k + R(i^*, j^*)$.

Para cada $(i, j) \in CS(k)$, $v_{ij}^k := v_{ij}^k + R(i^*, j^*)$ e $R(i, j) := R(i, j) - R(i^*, j^*)$.

Passo 3.

Atualizamos o dígrafo auxiliar fazendo:

$$E' := E' \cup \{(i^*, j^*)\}$$

e voltamos ao passo 1.

O algoritmo começa com uma solução dual viável com todas as variáveis com valor 0. Como a função objetivo dual é maximizar $\sum_{k \in V_0} u_k^k$, a estratégia de Wong consiste em incrementar os u_k^k enquanto se modificam as outras variáveis duais de maneira que a viabilidade dual seja mantida. Cada execução dos passos 1-3 tenta incrementar um u_k^k particular. No início, todos os elementos de V_0 são componentes enraizadas e assim todos os $u_k^k, k \in V_0$ são candidatos a ser incrementados no passo 1. O passo 1 escolhe arbitrariamente uma componente enraizada de D' e um elemento $k \in V_0 \cap T$. O passo 2 incrementa cada u_h^k, $h \in C(k)$, incluíndo a variável u_k^k da função objetivo. Cada variável $R(i,j)$ representa a folga para à restrição correspondente ao arco (i,j). A segunda parte do passo 2 incrementa as variáveis v_{ij}^k e decrementa as variáveis de folga $S(i,j)$ para os arcos $(i,j) \in CS(k)$. Observamos que como $(i^*, j^*) \in CS(k)$, $R(i^*, j^*) := S(i^*, j^*) - S(i^*, j^*) = 0$. Assim, o passo 3 sempre adiciona um arco (i^*, j^*) a D' cuja variável de folga foi a zero (ou cuja restrição $\sum_{k \in V_0} v_{ij}^k \leq w_{ij}$ correspondente é justa).

Como cada execução do passo 3 aumenta o conjunto E, o número de componentes enraizadas diminuirá. Portanto, na medida que o algoritmo avança, alguns u_k^k não serão incrementados.

Para mostrar o funcionamento do algoritmo ascendente dual, consideremos o seguinte dígrafo ponderado $D = (V, E, w), r = 1$, $V_0 = \{2,3,4,5\}$ e $V_f = \{6,7,8\}$ (fig.II.1). No início, todos os u_i^k e v_{ij}^k são zero.

A tabela II.1 mostra os detalhes do algoritmo ascendente para o exemplo.

Depois de acabar, a função objetivo dual é : $\sum_{k=2}^{5} u_k^k = 5+1+9+7 = 22$.

A arborescência ótima de Steiner dada por $\{(1,7), (7,4), (4,3), (4,2), (1,8), (8,5)\}$ também tem um peso total de 22 e portanto não existe um *'gap'* dual. Wong também prova que o algoritmo dual mantém em todo momento uma solução

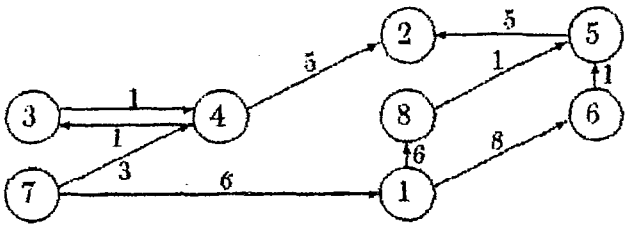


Figura II.1: Exemplo para ilustrar o algoritmo ascendente dual.

Iteração	T	$CS(k)$	$[i^*, j^*, R^*(i^*, j^*)]$	Variáveis Modificadas
1	{ 2 }	2	$\{(4,2),(5,2)\}$	$u_2^2 = v_{42}^2 = v_{52}^2 = 5$
2	{ 3 }	3	$\{(4,3)\}$	$u_3^3 = v_{43}^3 = 1$
3	{ 4 }	4	$\{(3,4),(7,4)\}$	$u_4^4 = v_{34}^4 = v_{74}^4 = 1$
4	{ 5 }	5	$\{(6,5),(8,5)\}$	$u_5^5 = v_{65}^5 = v_{85}^5 = 1$
5	{ 3,4 }	4	$\{(7,4)\}$	$u_4^4 = v_{74}^4 = 3,\ u_3^4 = 2$
6	{ 5 }	5	$\{(8,5),(1,6)\}$	nenhuma
7	{ 3,4 }	4	$\{(1,7)\}$	$u_4^4 = 9,\ u_3^4 = 8,\ u_7^4 = v_{17}^4 = 6$
8	{ 5 }	5	$\{(1,6),(1,8)\}$	$u_5^5 = 7,\ u_8^5 = u_8^6 = v_{18}^5 = v_{16}^5 = 6$

Tabela II.1: Operações do algoritmo ascendente dual para $D = (V, E, w)$.

viável para (DLP_2). Ele aproveita o limite inferior do PSD para encontrar soluções viáveis (e limites superiores) para o problema. A descrição do algoritmo completo é a seguinte:

1. Usar o procedimento ascendente dual para obter um limite inferior para o valor da solução ótima.

2. Considerar o dígrafo auxiliar D' quando o algoritmo ascendente acaba, sendo Q o conjunto de todos os vértices conectados à raiz 1. Notar que o vértice 1 e todos os elementos de V_0 pertencerão a Q (o algoritmo não terminará a menos que todos os elementos de V_0 estejam conectados ao vértice 1). Também alguns elementos de V_f estarão em Q.

3. Formar um problema de arborescência geradora mínima com o conjunto de vértices Q. O conjunto de arcos incluirá qualquer arco $(i,j) \in E$, onde i e j estão em Q. Os pesos w_{ij} são os pesos originais. Solucionamos o problema de arborescência geradora com o algoritmo ascendente e um procedimento para recuperar a arborescência ótima do dígrafo auxiliar D'.

4. Seja T^* a arborescência geradora mínima encontrada no passo 3. Se o vértice $v \in V_f$ é uma folha de T^* então eliminamos o vértice v e o arco entrando nele. Repetimos este processo até que nenhuma mudança adicional seja precisa. Os arcos restantes em T^* são uma solução viável para o PSD já que cada vértice em V_0 estará conectado ao vértice raiz. O peso desta solução viável é um limite superior para o valor da solução ótima.

O objetivo dos passos 2 e 3 é fazer com que a solução ótima para o PSD seja uma arborescência geradora mínima para o conjunto de vértices Z que são incidentes aos arcos na solução ótima. Caso contrário, a arborescência geradora mínima para Z daria uma solução de peso menor.

A idéia é 'adivinhar' quais vértices pertencem a Z. Claramente o vértice $r = 1$ e cada elemento de V_0 devem pertencer a Z. Também alguns elementos de V_f estarão em Z. Estes elementos de V_f serão aqueles que sejam úteis na tarefa de conectar elementos de V_0 ao vértice raiz 1 com peso mínimo. Para identificar estes vértices é usado o dígrafo auxiliar D' que resulta do procedimento ascendente dual. O conjunto Q é a aproximação do conjunto Z.

O passo 4 elimina arcos desnecessários da solução dada no passo 3 não afetando a viabilidade da solução.

Wong fez experimentos computacionais com seu algoritmo através de uma série de 24 problemas gerados aleatoriamente. Cada problema teste foi gerado na seguinte forma. Primeiro, uma árvore geradora não dirigida para o conjunto completo de vértices foi gerada. Depois, arestas adicionais foram aleatoriamente geradas até que a quantidade desejada estivese presente. Para converter a instância não dirigida anterior numa dirigida, Wong trocou cada aresta $[i, j]$ por dois arcos que são (i, j) e (j, i), os dois com o mesmo peso w_{ij}.

Os pesos dos arcos w_{ij} foram aleatoriamente selecionados do intervalo $(0, 1)$ e a quantidade de vértices em V_f foi escolhida como sendo $\frac{|V|}{2}$.

A Tabela II.2 mostra os resultados da experiência computacional do algoritmo de Wong.

Vértices	Arcos	Problemas Testes	Tempo Médio CPU	Problemas Resolvidos no Ótimo	$\frac{Limite Superior}{Limite Inferior}$
40	120	6	0.3783	6	—
40	160	6	0.4020	5	1.0071
60	180	6	0.5678	6	—
60	240	6	0.5875	5	1.0049

Tabela II.2: Resumo da experiência computacional do algoritmo de Wong.

Os resultados obtidos por Wong são excelentes não entanto temos duas considerações sérias para esses resultados.

(i) Pouca experiência computacional. Se testaram apenas dígrafos com 40 e 60 vértices e com densidade baixa ($2 \mid V \mid$ e $3 \mid V \mid$).

(ii) A construção do dígrafo foi feita de uma forma muito particular, já que dados os vértices i e j se existe o arco (i, j) também existe o arco (j, i). Na verdade, esta construção corresponde à classe particular dos dígrafos chamados **bidirecionados.**

II.3 Outras Aproximações

Comentaremos aqui outros trabalhos importantes dedicados ao PSD.

O PSD foi definido pela primeira vez (segundo nosso conhecimento) por Hakimi [24]. No entanto, o primeiro trabalho específico sobre o PSD é aquele de Nastansky, Selkow e Stewart [34]. Eles apresentam um algoritmo de enumeração implícita para encontrar a solução ótima do PSD no caso particular de dígrafos sem circuitos; algumas aplicações também são comentadas. L. Guyard [23] na sua tese de doutorado modelou a implementação ótima de uma relação numa base de dados relacional para um sistema de projeções através do PSD. Palma-Pacheco [33] também na sua tese de doutorado dá um algoritmo heurístico usando duas etapas; a primeira etapa considera a solução dada pelo algoritmo de Wong para obter um limite inferior e uma solução viável que representa um limite superior. A segunda etapa tenta melhorar a solução viável. Ele dá resultados computacionais mostrando a qualidade da heurística. Chopra e Rao [9] dão formulações de programação linear inteira para

as versões dirigida e não dirigida do PSD, e estudam os poliedros associados. Liu [29] introduz uma nova formulação de programação inteira para o problema; esta formulação está baseada numa formulação de programação linear para o poliedro da **árvore de Steiner de dois terminais** obtida por Ball e outros [3]. Liu dá uma forma geral do algoritmo ascendente dual que ele aplica à nova formulação, para obter um limite inferior para o PSD. No algoritmo, ele usa o algoritmo ascendente dual introduzido por Wong [42] como uma subrotina e melhora o limite inferior

Capítulo III

Testes de Redução para o Problema

III.1 Reduções para o caso não direcionado

Os testes de redução representam um importante primeiro passo na solução exata de problemas de otimização combinatória. Experiências de testes de redução existem para vários problemas; entre eles, para os problemas da Mochila, ver [26]; Localização, ver [1] e Steiner, ver [2,4,13,14,15,38,41].

O objetivo dos testes consiste em reduzir o tamanho do espaço de busca fixando variáveis. Em particular, para o PSG os principais testes permitem identificar

- arestas e vértices facultativos que devem pertencer à solução ótima.
- arestas e vértices facultativos que não pertencem à solução ótima.

Algumas reduções também são possíveis de aplicar durante o percurso de um algoritmo de enumeração exata para o problema.

Estudemos primeiro algumas reduções simples obtidas de apenas propriedades locais do grafo.

- Se G contém um vértice facultativo de grau 1 então tal vértice pode ser eliminado do grafo.

- Se G contém um vértice facultativo v de grau 2 então podemos eliminar v do grafo e acrescentar uma nova aresta ligando seus vizinhos. O peso da nova aresta é a soma dos pesos das antigas arestas incidindo em v.

- Se G contém um vértice obrigatório v de grau 1 então podemos eliminá-lo do grafo, acrescentando o vértice adjacente a v ao conjunto de vértices obrigatórios V_0.

- Se G contém uma aresta $[u, v]$ tal que $d[u, v] < w[u, v]$ então ela pode ser eliminada do grafo. Se $d[u, v] = w[u, v]$ e existe um caminho de u a v que não utiliza a aresta $[u, v]$, também podemos eliminá-la do grafo. $d[u, v]$ denota o peso do caminho mais curto entre u e v. Chamaremos este teste, **Teste do Custo Mínimo** (TCM).

Vejamos agora outros testes menos simples. Suporemos que os pesos w_{uv} das arestas são todos distintos. Esta não é uma hipótese restritiva.

- Seja $v \in V_0$ um vértice obrigatório de G, u o vértice mais próximo de v e z o segundo vértice mais próximo de v. Se

$$w[u, v] + min_{x \in V_0 - u}\{d[u, x]\} \leq w[v, z]$$

então a aresta [u,v] está na solução ótima e assim podemos contraí-la em G.

Este teste pode ser generalizado, ver Duin e Volgenant [13], da seguinte forma:

- Dado um grafo G e uma aresta $e = [x, y]$, se existem vértices $u, v \in V_0$ tais que

$$d[u, v] = d[u, x] + w(e) + d[y, v]$$

e todo caminho entre u e v em $G - e$ contém uma aresta f tal que

$$w(f) \geq d[u, v]$$

então e está na solução ótima e pode ser contraída.

- Se G contém 2 vértices obrigatórios adjacentes u e v e existe vértice obrigatório z, $u \neq z$, e $v \neq z$ tal que $w[u,v] \geq d[u,z]$ e $w[u,v] \geq d[v,z]$, então $[u,v]$ pode ser eliminada do grafo.

Analisemos agora reduções que, para grafos densos, não dependem da função de pesos para diminuir os tamanhos das instâncias. São 4 testes os quais identificam arestas que necessariamente devem pertencer ou ser excluídas da solução ótima. Para qualquer subconjunto de vértices $V' \subseteq V$, o **subgrafo induzido por** V', que escreveremos $G[V']$, consiste de todos os vértices em V' e todas as arestas de G que tem ambos vértices terminais em V'. Denotaremos por $T(V')$ a árvore geradora mínima de $G[V']$. As provas das proposições seguintes aparecem em Balakrishnan e Patel [2].

Proposição III.1. Para qualquer par de vértices $u, v \in V_0 \cup V_f^*$, se a aresta [u,v] pertence a $T(V)$ (a árvore geradora mínima de G) então deve pertencer também a T^*.

A proposição anterior supõe que o conjunto V_f^* de pontos de Steiner gerados pela árvore geradora mínima T^* é conhecido. No entanto, esta informação não está disponível até que o problema esteja completamente resolvido. Em qualquer caso, a proposição pode ser usada na seguinte forma:

Cada aresta $[u,v]$ de $T(V)$, $u, v \in V_0$, deve pertencer a T^*. Logo cada vez que dois vértices obrigatórios forem adjacentes em $T(V)$, eles podem ser substituídos por um único vértice equivalente. Se aparecer arestas paralelas neste processo, somente a aresta de peso menor necessita ser preservada.

Adicionalmente, suponhamos que a árvore geradora mínima $T(V)$ contém uma aresta $[u,v]$, $u \in V_f$ e $v \in V_0$. Então a proposição III.1 estabelece que T^* deve conter a aresta $[u,v]$ se ela gera o vértice de Steiner u.

Proposição III.2. T^* não contém qualquer aresta $[u,v]$, $u, v \in V_0$ que não pertença a $T(V_0)$.

Esta proposição traz como consequência que, de todas as arestas $[u,v]$, $u, v \in V_0$, somente as $(\mid V_0 \mid -1)$ arestas que pertencem a $T(V_0)$ preci-

sam ser retidas. Para grafos completos, esta redução diminui o número de arestas $[u, v], u, v \in V_0$ a $2/\mid V_0 \mid$ vezes o tamanho original.

Proposição III.3. T^* contém uma aresta $[u, v], u \in V_f$ e $v \in V_0$, se e somente se esta aresta pertence a $T(V_0 + u)$.

Proposição III.4. T^* contém a aresta $[u, v]$, $u, v \in V_f$, somente se $T(V_0 \cup \{u, v\})$ contém esta aresta.

Estes últimos dois testes reduzem o tamanho do problema eliminando arestas incidentes aos vértices facultativos. Na aplicação da proposição III.3, precisamos encontrar, para cada vértice facultativo u, a árvore geradora mínima $T(V_0+u)$ do subgrafo induzido $G[V_0 + u]$. Se, para algúm vértice v em V_0, a aresta $[u, v]$ pertence ao grafo original G mas não a $T(V_0 + u)$, então esta aresta pode ser eliminada. O teste baseado na proposição III.4 precisa no máximo de $\mid V_0 \mid (\mid V_0 \mid -1)/2$ soluções de árvore geradora mínima $T(V_0 \cup \{u, v\})$, uma para cada par de vértices facultativos u, v.

Os testes anteriores também podem ser aplicados em estágios intermédiarios de procedimentos de enumeração usados para resolver o PSG. Adicionalmente, os distintos testes podem ser usados de maneira iterativa, ou seja, a redução atingida usando um teste poderia servir para aumentar a redução aplicando outros testes. Estas e outras propriedades podem ser vistas em detalhe no trabalho de Balakrishnan e Patel [2]. Eles fazem implementação computacional e dão resultados mostrando a qualidade dos testes.

III.2 Reduções Propostas para o Caso Direcionado

Daremos nesta seção duas classes de testes de redução para o PSD. A primeira classe inclui testes que são uma extensão direta dos testes conhecidos para o PSG. A segunda classe está composta de testes novos especialmente desenhados para o caso não simétrico de grafos direcionados. Adicionalmente, provamos que alguns testes importantes para o PSG não podem ser estendidos ao caso direcionado.

Grau Externo Zero. Cada vértice $v \in V_f$ com $d^+(v) = 0$ pode ser eliminado. Ver figura III.1.

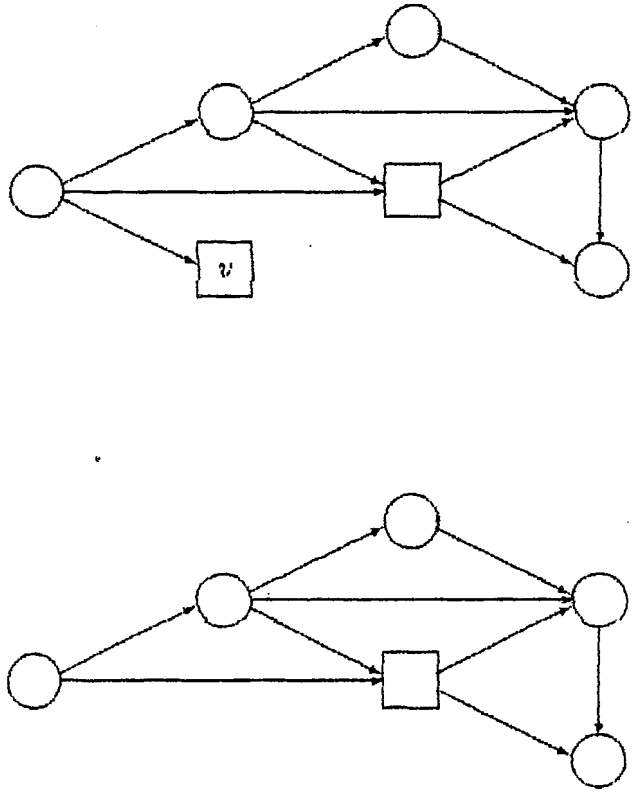
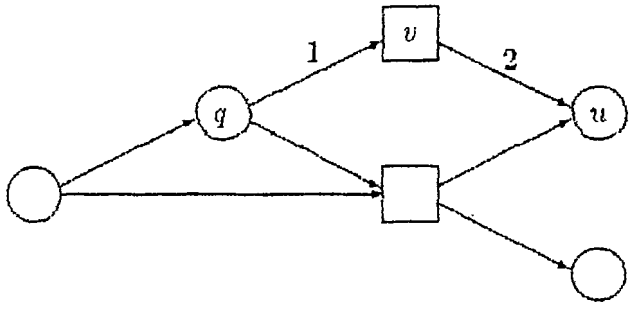


Figura III.1: Teste do Grau Externo Zero.

Grau Externo Um. Cada vértice $v \in V_f$ com $d^+(v) = 1$ pode ser eliminado. Se o arco existente é o arco (v, u), $u \in V - v$ então eliminamos o vértice v criando para cada arco (q, v) um arco (q, u) com peso $(w_{qv} + w_{vu})$. Ver figura III.2.



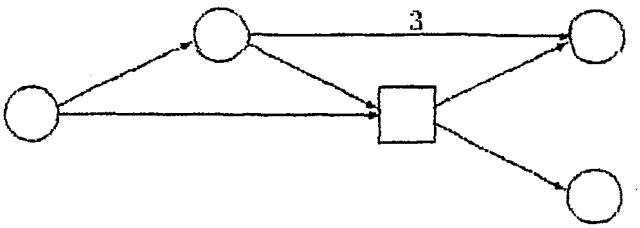


Figura III.2: Teste do Grau Externo Um.

No caso de aparecer arcos paralelos durante a aplicação do teste, é suficiente manter aquele arco de peso menor.

Grau Interno Um. Se o vértice $v \in V_0$ tem $d^-(v) = 1$ então o arco (u, v), $u \in V - v$ deve estar em qualquer solução ótima e logo os vértices u e v podem ser fundidos. Se $u \in V_f$ então depois da fusão $u \in V_0$. Ver figura III.3.

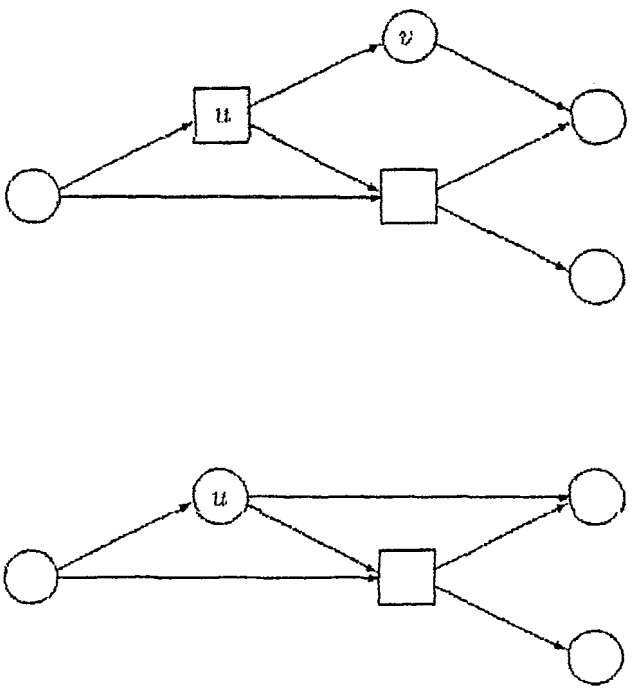


Figura III.3: Teste do Grau Interno Um.

O seguinte teste usa a matriz $[d_{uv}]_{u,v \in V}$, de distâncias mais curtas

com respeito à matriz de pesos originais w_{uv}, onde $w_{uv} = \infty$ se $(u, v) \notin E$.

Se $P_{uv} = \{u = u_0, u_1, \ldots, u_n, v = u_{n+1}\}$ é um caminho direcionado de u para v em D que possui a distância mais curta entre u e v em D então $d_{uv} = \sum_{k=0}^{n} w_{u_k u_{k+1}}$ e evidentemente satisfaz $d_{uv} \leq w_{uv}$. Agora podemos formular o teste seguinte.

Lema III.1. Teste do Custo Mínimo (TCM).

O arco (u, v), $u, v \in V$ pode ser eliminado se $d_{uv} < w_{uv}$.

Prova. Seja $P_{uv} = \{u = u_0, u_1, \ldots, u_n, v = u_{n+1}\}$ um caminho de u a v que satisfaz $d_{uv} < w_{uv}$. Note que qualquer solução T que inclua o arco (u, v) pode ser transformada numa outra r-arborescência T' que não inclui (u, v) e que tem custo menor. Para isto procuramos o vértice u_l em P_{uv} que tem máximo índice l que pertence ao caminho de r a u em T; inserimos em T a parte $\{u_l, u_{l+1}, \ldots, v\}$ de P_{uv}, eliminando cada arco (se existe) entrando em cada vértice u_k, $l + 1 \leq k \leq v$. A solução assim obtida T' deve ter um custo menor que T já que não existem arcos com pesos negativos. ∎

Duin e Volgenant [14] propõem um teste de eliminação de arestas para o PSG, chamado o **Teste da Distância Especial** (TDE), o qual generaliza os 3 testes anteriores dados pelas proposições III.2–3–4. O poder de poda do TDE, segundo os resultados obtidos, é altíssimo. Veremos contudo que o TDE não tem uma extensão possível a dígrafos. Isto acontece porque as versões em dígrafos das proposições anteriores também não são válidas.

Proposição III.2'. T^* não contém qualquer arco (u, v), $u, v \in V_0$ que não pertença a $T(V_0)$.

Contraexemplo. Consideremos a instância do PSD mostrada na figura III.4. Seja $u = u_2$ e $v = u_3$. Temos que T^* está dada pela figura III.5 e $T(V_0)$ está dada na figura III.6. Observamos que $(u, v) \notin T(V_0)$ e no entanto $(u, v) \in T^*$. Assim, a proposição III.2' é falsa.

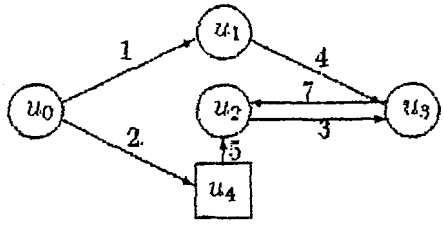


Figura III.4: Dígrafo Inicial.

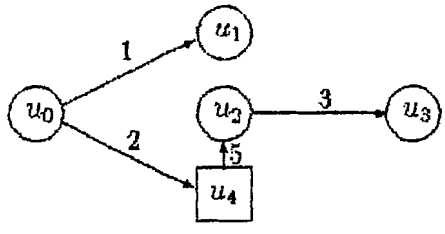


Figura III.5: $w(T^*) = 11$.

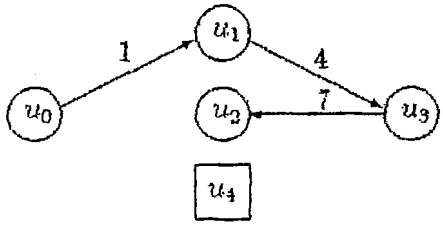


Figura III.6: $w(T(V_o)) = 12$.

Proposição III.3'. T^* contém um arco (u, v), $u \in V_f$, $v \in V_0$ somente se $(u, v) \in T(V_0 + u)$.

Contraexemplo. Consideremos a instância do PSD dada na figura III.7. Seja $u = u_3$ e $v = u_1$. Então $T(V_0 + u)$ é dada na figura III.8 e T^* é dada na figura III.9. Notamos que $(u, v) \notin T(V_0 + u)$ mas $(u, v) \in T^*$. Assim a proposição III.3' é falsa.

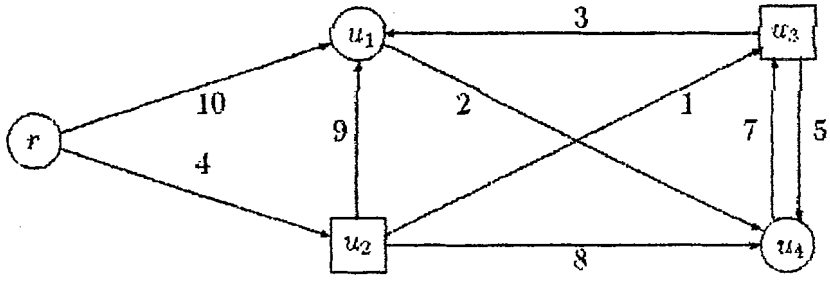


Figura III.7: Dígrafo Inicial.

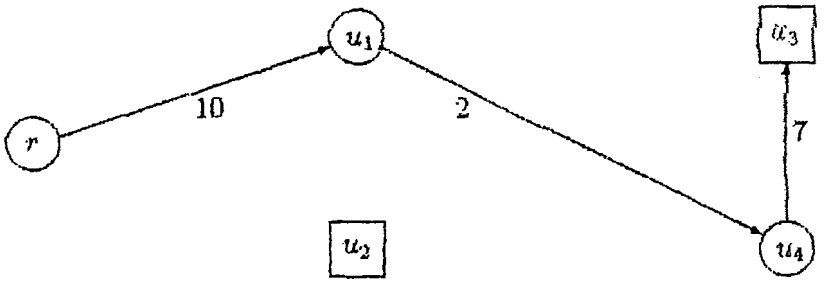


Figura III.8: $w(T(V_o \cup \{u\})) = 19$.

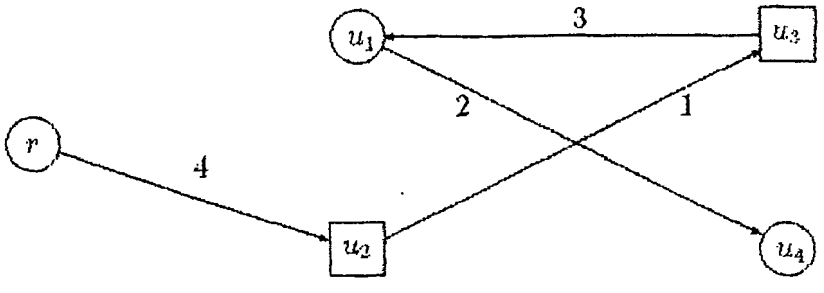


Figura III.9: $w(T^*) = 10$.

Proposição III.4'. T^* contém o arco (u, v), $u, v \in V_f$ somente se $T(V_0 \cup \{u, v\})$ contém o arco (u, v).

Contraexemplo. Consideremos a seguinte instância do PSD dada na figura III.10.

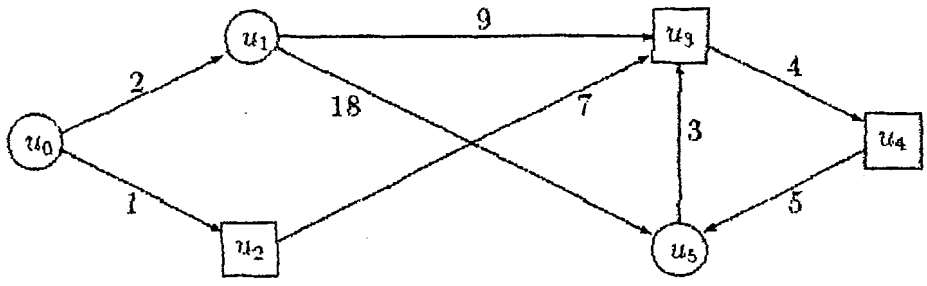


Figura III.10: Dígrafo Inicial.

Seja $u = u_2$ e $v = u_3$. Então $T(V_0 \cup \{u, v\})$ está dada na figura III.11.

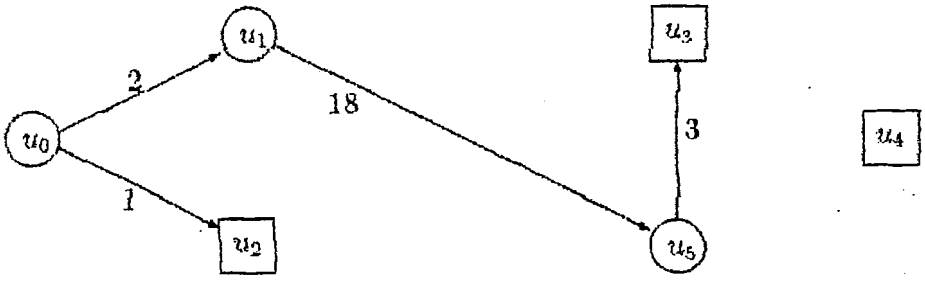


Figura III.11: $w(T \cup \{u, v\}) = 24$.

e T^* está dada na figura III.12.

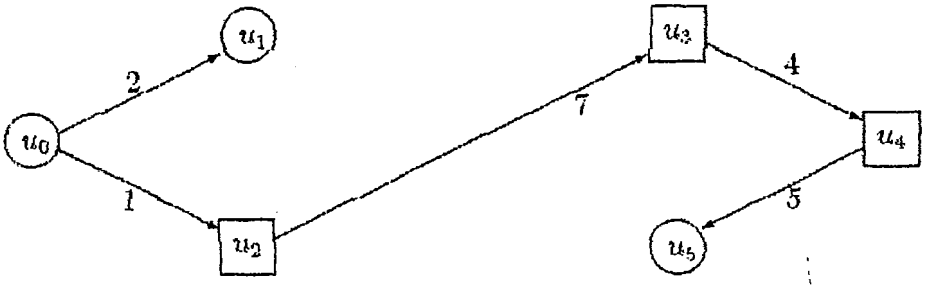


Figura III.12: $w(T^*) = 19$.

Notamos que o arco $(u, v) \notin T(V_0 \cup \{u, v\})$ mas $(u, v) \in T^*$. Portanto a proposição 4' é falsa e o arco (u, v) não pode ser eliminado.

Desenvolveremos agora três testes que consideram as direções dos arcos.

O primeiro teste considera apenas a topologia do dígrafo independendo assim da estrutura da função de pesos.

Lema 2. Teste do Vértice Corte (TVC).

Sejam $u, v \in V$ vértices tais que cada caminho da raiz r ao vértice u inclui o vértice v. Então o arco (u, v) é redundante em D.

Prova. Seja T uma r-arborescência de Steiner em D. Se (u, v) pudesse participar de T, então para evitar circuitos, não existiriam caminhos de r a u em T e logo não seria uma r-arborescência de Steiner. ∎

Consideremos a instância do PSD dada pela figura III.13

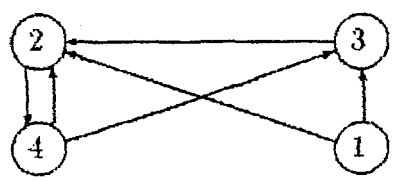


Figura III.13: Teste do Vértice Corte.

Observamos que todo caminho de $r = 1$ ao vértice 4 inclue o vértice 2, logo o arco $(4, 2)$ é redundante em D.

Lema 3. Teste da Raiz (TR).

O arco (u, r), $u \in V - r$ pode ser eliminado.

Observamos que o TR é um caso particular do TVC que acontece quando no TVC fazemos $v = r$.

Temos um último teste, o qual faz uso da matriz $[d_{uv}]_{uv \in V}$. Este teste também permite eliminar arcos redundantes em D. $S(u)$, $u \in V$, denotará o conjunto de vértices sucessores do vértice u em D. Logo, $v \in S(u)$ se e somente se existe caminho de u a v em D.

Lema 4. Teste do Vizinho mais Próximo (TVP).

O arco (u,v), $u,v \in V$ pode ser eliminado em D se existe um arco (k,v) em D, $k \in \{V_0 \cup \{r\}\} - S(v)$ tal que $w_{kv} < w_{uv}$.

Prova. Seja T uma r-arborescência de Steiner de D que contém o arco (u,v) e suponhamos que D contém um arco (k,v) com $k \in \{V_0 \cup \{r\}\} - S(v)$ e $w_{kv} < w_{uv}$. Se eliminamos o arco (u,v) em T então separamos T em duas subarborescências de D, T_u e T_v, onde T_u é uma r-arborescência de T que satisfaz $d_{rv} = \infty$, e T_v é uma j-arborescência de T.

Como $k \in \{V_0 \cup \{r\}\} - S(v)$ então $k \in T_u$ (de outra maneira, a existência de (k,v) em D implicaria $k \in S(v)$) e logo $T' = T - \{(u,v)\} \cup \{(k,v)\}$ é uma r-arborescência de Steiner em D com $w(T') = w(T) - w_{uv} + w_{kv}$. Por hipótese, $w_{kv} - w_{uv} < 0$ implica $w(T') < w(T)$ negando o fato que T é mínima. ∎

É interessante notar que se no TVP não fosse exigida a condição $k \notin S(v)$ então, no caso particular $V = V_0$, verificaríamos que uma estratégia gulosa resolveria o problema de encontrar uma r-arborescência geradora mínima em D, o qual sabemos não é verdade, ver Edmonds [16].

Consideremos a instância do PSD dada pela seguinte figura III.14.

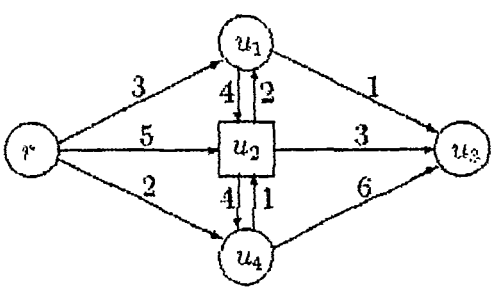


Figura III.14: Instância do PSD para ilustrar o TVP.

A figura III.15 ilustra uma sequência de eliminação de arcos aplicando o TVP. Observamos do dígrafo final que as duas soluções ótimas T_1 e T_2 podem ser obtidas. Elas são apresentadas na figura III.16.

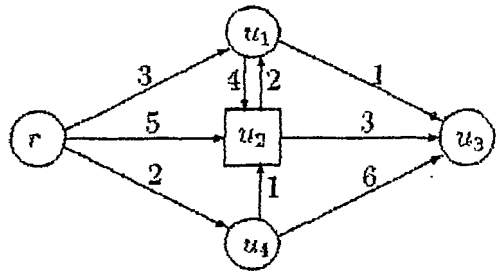
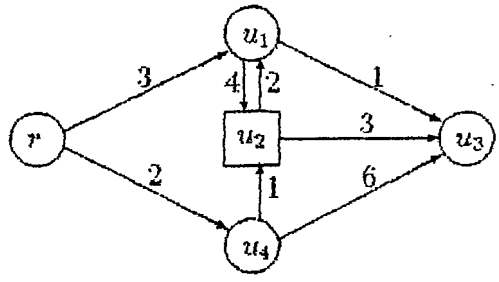
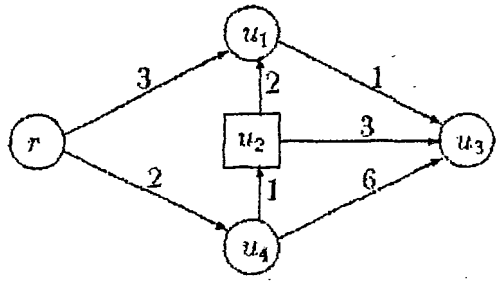
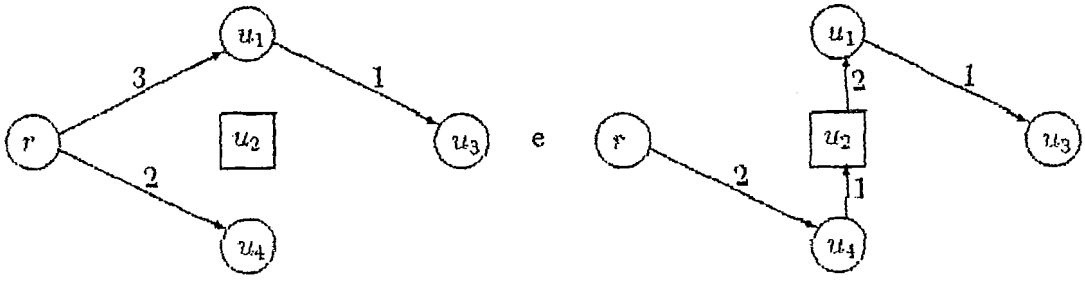
Eliminação	Dígrafo depois da eliminação
(u_2, u_4) Existe $(r, u_4) \in E$, $r \in \{r\} - \mathcal{S}(u_4)$ com $w_{ru_4} < w_{u_2u_4}$	
(r, u_2) Existe $(u_4, u_2) \in E$, $u_4 \in V_o - \mathcal{S}(u_2)$ com $w_{u_4u_2} < w_{ru_2}$	
(u_1, u_2) Existe $(u_4, u_2) \in E$, $u_4 \in V_o - \mathcal{S}(u_2)$ com $w_{u_4u_2} < w_{u_1u_2}$	

Figura III.15: Eliminação de arcos pelo TVP.



$w(T_1) = 6$ $w(T_2) = 6$

Figura III.16: As duas soluções da instância dada.

III.3 Complexidade Computacional dos Testes

Analisaremos a complexidade computacional dos testes TCM, TVP e TVC. A implementação dos outros testes é trivial.

No caso do teste TCM, precisamos da matriz $[d_{uv}]_{u,v\in V}$, dos caminhos mais curtos entre todos os pares de vértices com respeito à matriz de pesos originais $[w_{uv}]_{u,v\in V}$. O conhecido algoritmo de Floyd [19] resolve este problema em tempo $O(\mid V \mid^3)$. O trabalho posterior é de apenas uma comparação por arco($d_{uv} < c_{uv}$), logo o TCM tem complexidade total $O(\mid V \mid^3 + \mid E \mid) = O(\mid V \mid^3)$.

No teste TVP, para cada arco (u, v) devemos saber se existe um outro arco l que tenha um peso menor que o peso de (u, v) e $d_{vl} = \infty$. Como existem $O(\mid V_0 \mid)$ de tais vértices l, a complexidade total do TVP é $O(\mid V_0 \mid\mid E \mid)$ supondo que $[d_{uv}]_{u,v\in V}$ já foi calculada.

Para o TVC faremos a seguinte implementação. Para o arco (u, v) eliminamos todos os arcos l entrando no vértice v fazendo $w_{lv} = \infty$. Depois calculamos a distância de r a u e eliminamos o arco (u, v) se $d_{ru} = \infty$. Finalmente, atualizamos os custos w_{uv} para estudar um novo arco. Se o cálculo da distância de r a i é calculada aplicando o algoritmo de Dijkstra [12] que é da ordem $O(\mid V \mid^2)$ então a complexidade total do teste é $O(\mid V \mid^2 \mid E \mid)$.

Portanto, fica provado que os testes podem ser implementados em tempo polinomial.

III.4 Resultados Numéricos e Conclusões

Implementamos computacionalmente cinco testes, o TR, o TGZ, o TCM, o TVP, e o TVC. Para a experimentação computacional geramos aleatoriamente dígrafos de diferentes tamanhos e densidades.

Primeiro, para garantir a existência de soluções viáveis para uma instância dada do PSD geramos arcos na seguinte forma: criamos dois conjuntos

V_1 e V_2. No inicio, $V_1 = r$ e $V_2 = V - V_1$, depois escolhemos aleatoriamente dois elementos $u, v, u \in V_1, v \in V_2$; os vértices u e v definem o arco (u, v). Os conjuntos V_1 e V_2 são atualizados como $V_1 \longleftarrow V_1 + v$ e $V_2 \longleftarrow V_2 - v$. Após $\mid V \mid -1$ passos obtemos uma r-arborescência geradora para $D = (V, E)$ e a seguir geramos arcos adicionais até que o número desejado de arcos seja obtido. Os pesos dos arcos w_{uv} foram inteiros aleatoriamente gerados de dois tamanhos distintos de intervalos, $w_1 = [1, 10]$ e $w_2 = [1, 50]$.

Para cada instância aleatoriamente gerada na forma já descrita, aplicamos repetidamente (salvo o TR) os 5 testes definidos anteriormente, na ordem dada, até que nenhuma redução adicional fosse obtida. Foram programadas séries de 10 problemas de cada tamanho de pesos w_1 e w_2. A Tabela III.1 na página 36 dá os resultados obtidos da experiência computacional; estes resultados são médias sobre os 10 problemas arredondados a valores inteiros e são acumulativos, ou seja, definem a redução do teste para a instância resultante depois da aplicação dos testes anteriores. As reduções obtidas são significativas e discutiremos detalhadamente a suas contribuções individuais.

O poder de poda do TR, como é lógico, depende apenas da quantidade de arcos na instância dada e podemos observar que elimina da ordem de 1% dos arcos.

Para o teste TGZ acontece o contrário, ou seja, quando aumenta a densidade do dígrafo decresce o poder de poda; claramente num dígrafo denso a probabilidade de encontrar um vértice com $d^+(v) = 0$ é muito pequena.

Com respeito ao TCM observamos que o teste aumenta o poder de poda com a densidade do dígrafo; esta relação se justifica pelo fato de existir mais caminhos entre cada par de vértices num dígrafo com maior número de arcos.

O teste TVP tem características parecidas ao TCM e na Tabela III.1, como já foi comentado, aparece apenas o efeito do teste depois da aplicação dos testes TR, TGZ, e TCM.

Finalmente, observamos que o teste TVC é eficaz apenas para instâncias

com baixa densidade e se recomenda usá-lo apenas nesse caso.

$\|V\|$	$\|V_0\|$	$\|E\|$	TR		TGZ		TZM		TVP		TVC		TOTAL	
			w_1	w_2	w_1	w_2	w_1	w_2	w_1	w_2	w_1	w_2	w_1	w_2
50	10	100	1	2	23	18	1	2	7	8	9	11	41	54
		250	4	5	2	4	26	34	15	13	1	1	48	59
		500	11	10	0	1	171	243	23	16	0	0	205	270
		1500	30	31	0	0	997	1182	19	16	0	0	1052	1232
	25	100	2	1	18	13	1	2	12	13	7	5	40	34
		250	3	6	1	2	27	42	13	15	0	1	44	66
		500	10	10	0	0	176	236	25	17	0	0	211	261
		1500	28	31	0	0	972	1179	43	18	0	0	1043	1229
	40	100	1	1	6	5	2	1	34	35	2	2	45	44
		250	5	3	1	3	22	47	47	37	0	1	75	91
		500	8	8	0	0	167	251	27	20	0	0	202	279
		1500	31	28	0	0	979	1195	39	16	0	0	1049	1238
100	25	200	2	2	40	34	2	1	12	14	24	20	80	72
		500	4	6	2	1	29	58	17	17	0	2	53	84
		1000	10	11	0	1	268	416	29	20	0	0	307	447
		4000	38	39	0	0	2669	3261	57	25	0	0	2764	3325
	50	200	2	1	29	27	1	3	22	24	14	14	68	69
		500	5	4	2	3	32	66	17	16	2	2	58	91
		1000	12	8	0	0	271	431	34	21	0	0	317	460
		4000	40	41	0	0	2659	3250	82	25	0	0	2781	3316
	75	200	1	1	9	14	2	2	26	38	8	6	46	61
		500	5	4	2	0	32	55	19	22	1	1	57	82
		1000	10	9	0	0	269	424	30	29	0	0	308	462
		4000	38	41	0	0	2661	3258	72	23	0	0	2771	3322

Tabela III.1: Resultados Numéricos dos Testes de Redução. $w_1 = [1, 10]$, $w_2 = [1, 50]$.

Com respeito à influência dos tamanhos dos pesos podemos observar qur no teste TCM a potência do teste aumenta junto com o tamanho do intervalo. Esta observação também foi feita por Winter e Mc Gregor Smith [41].

De uma forma geral, podemos afirmar que o conjunto de testes de redução dado oferece uma efetiva etapa de preprocessamento do dígrafo ainda no caso de instâncias esparsas.

Para futuras pesquisas sugerimos fazer experimentos com um número maior de intervalos para os pesos dos arcos. Também parece importante estudar testes efetivos para eliminar vértices facultativos

Capítulo IV

Um Algoritmo Exato para o Problema

IV.1 Idéia do Algoritmo

A idéia central do algoritmo proposto consiste em solucionar uma formulação de arborescência geradora mínima restrita para o PSD. Esta idéia foi originalmente sugerida, mas não testada computacionalmente, para o PSG por Balakrishnan e Patel [2]. Tal formulação foi explicitada através de Programação Linear Inteira (PLI) por Beasley [5]. Ele aplica a tal programa inteiro um procedimento de Relaxação Lagrangeana que gera limites inferiores num algoritmo Branch and Bound obtendo resultados numéricos bons para instâncias de tamanho grande. No entanto, a formulação não precisa de tal explicitação; é possível solucionar a formulação através de um esquema que identifica a arborescência ótima listando arborescências geradoras de um dígrafo aumentado na ordem crescente de pesos. O esquema se fundamenta no fato que qualquer problema de arborescência geradora mínima restrita pode ser resolvido listando arborescências do dígrafo na ordem crescente de pesos, até identificar a arborescência mínima que satisfaz as restrições adicionais.

IV.2 Algoritmo de Ranking para Arborescências Geradoras

O algoritmo de ranking de arborescências geradoras para um dígrafo ponderado $D = (V, E, w)$, que descreveremos aqui é aquele de Camerini, Fratta e Maffioli [7]. Faremos apenas leves modificações para adaptar tal algoritmo ao nosso caso de um problema de mínimo.

Seguindo o trabalho [7], denotaremos uma rede $R = (V, E, w)$ por $R = (V, E, \sigma, \tau, w)$, onde σ e τ são as funções de incidência de R. Para $e = (u, v) \in E$ denotaremos $\sigma(e) = u$ e $\tau(e) = v$. Especificaremos V por $V = \{1, 2, \ldots, n\}$ e suporemos que $\mid E \mid = m$. Um **Branching** B de R é uma subrede de R que não contém circuitos e no máximo um arco está direcionado a cada vértice. O vértice $v \in V$ está **exposto** com respeito a B se nenhum arco de B está direcionado para v. Se (u, v) representa um caminho C com apenas um arco então dizemos que u é o **pai** de v e que v é o **filho** de u. Uma arborescência geradora de uma rede pode ser também definida como um branching com uma única raiz. Suporemos que R não tem laços e que contém no mínimo uma arborescência geradora com raiz no vértice 1. Uma **melhor arborescência geradora** (MAG) para R será uma arborescência geradora (AG) de peso mínimo com raiz em 1.

Seja Y um branching arbitrário de R sem arcos direcionados para 1 e suponhamos que Y é um subconjunto de uma AG de R. Seja $Z \subseteq E$ tal que $Z \cap Y = \emptyset$. Uma subrede R_{YZ} de R é definida da seguinte maneira:

$$R_{YZ} = (V, E_{YZ}, \sigma, \tau, w), \text{ onde } E_{YZ} = E - (Z \cup \{e \in E - Y \mid \exists e' \in Y,\ \tau(e') = \tau(e)\}).$$

Uma propriedade direta de R_{YZ} é que A é uma (M)AG de R sujeita as restrições $Y \subseteq A \subseteq E - Z$ se e somente se A é uma (M)AG de R_{YZ}. Portanto, o problema de encontrar uma MAG sujeita às restrições anteriores é equivalente àquele de encontrar uma MAG sem restrições.

Suponhamos agora que existe um circuito C de R, onde $C = (V' =$

$\{v_1, v_2, \ldots, v_k = v_1\}$, $E' = \{e_1, e_2, \ldots, e_{k-1}\}$, $\sigma, \tau, w)$. Suponhamos também que C não contém o vértice 1 e que para cada h, $1 \leq h < k$, $w(e_h)$ é mínimo entre todos os arcos direcionados para v_{h+1}. A rede $R_c = (V_c, E_c, \sigma_c, \tau_c, w_c)$ chamada a **rede colapsada** em C é definida como segue. Sejam $u_1 = 1, u_2, \ldots, u_{n-k-1}$ os vértices de $V - V'$ e $V_c = \{u_1 = 1, u_2, \ldots, u_{n-k-2}\}$. O vértice artificial u_{n-k-2} é um inteiro maior que $max\{v \mid v \in V\}$, arbitrário e identifica um novo vértice que é chamado de vértice de R_c correspondente a C. Seja $E_c = E - \{e \in E \mid \sigma(e), \tau(e) \in V'\}$. Para cada $e \in E_c$ definimos:

$$\sigma_c(e) = \begin{cases} u_{n-k+2} & \text{,se } \sigma(e) \in V' \\ \sigma(e) & \text{, caso contrário} \end{cases}$$

$$\tau_c(e) = \begin{cases} u_{n-k+2} & \text{,se } \tau(e) \in V' \\ \tau(e) & \text{, caso contrário} \end{cases}$$

$$w_c(e) = \begin{cases} w(e) - w(e_h) & \text{,se } \tau(e) \in V' \\ w(e) & \text{, caso contrário} \end{cases}$$

onde e_h é o único arco de C que satisfaz $\tau(e_h) = \tau(e)$.

Seja A uma AG de R_c e e o único arco de A tal que $\tau(e) \in V'$. Suponhamos que adicionamos a A todos os arcos de C exceto e_h. O resultado é uma AG de R a qual podemos chamar de AG **expandida** de A (com respeito a C) e será denotada $X_c(A)$.

Lema IV.1. Sejam A_1, A_2 AG de R_c. Então

$$w_c(A_1) - w_c(A_2) = w(X_c(A_1)) - w(X_c(A_2)).$$

O segundo lema diz, em particular , que se A é uma MAG de R então ela necessariamente deve ser expandida por uma AG A' de R_c.

Lema IV.2. Seja A uma AG de R. Então existe uma AG A' de R_c tal que

$$w(X_c(A')) \leq w(A).$$

Lema IV.3. Seja $\overline{A}$ uma MAG de R_c. Então $X_c(A)$ é uma MAG de R.

O seguinte procedimento **melhor** implementa os lemas anteriores e dá como resultado uma MAG A de R sujeita as restrições $Y \subseteq A \subseteq E - Z$.

Procedure Melhor(Y,Z)
Begin
1. $M \leftarrow R_{Y,B}$; $B \leftarrow \emptyset$;
2. Defina um branching C sem arcos e com vértices $1, \ldots, n$;
3. **while** exista um vértice exposto $v \neq 1$ de M respeito de B **Do Begin**
4. Sejam σ_M, τ_M, w_M, as funções de incidência e peso de M;
5. Seja b um arco de M tal que $\tau_M(b) = v$ e $w_M(b)$ é máximo;
6. $B \leftarrow B \cup \{b\}$;
7. $\beta(v) \leftarrow b$;
8. **If** B contém um circuito C de M **Then Begin**
9. $M \leftarrow M_c$; seja u o vértice de M_c correspondente a C;
10. Adicione a C o novo vértice u;
11. **For** cada vértice W de C **Do** faça W um filho de U em C;
12. $B \leftarrow B - C$

End

End

13. **While** C contém vértices não-isolados **Do Begin**
14. Identifique em C o caminho com vértices $v_1, \ldots, v_k$;
onde v_1 é qualquer raiz não isolada de C e $v_k = \tau(\beta(v_1))$;
15. **For** $h \leftarrow 1$ **Until** $k - 1$ **Do Begin**
16. $\beta(v_{h+1}) \leftarrow \beta(v_h)$;
17. Elimine de C o vértice v_h e todos os arcos direcionados fora de v_h

End

End

18. $A \leftarrow \{\beta(v) \mid v \neq 1$ é um vértice de $R\}$;
Return A

End

O procedimento **melhor** pode ser dividido em duas fases. A primeira fase, Fase de **Contração** é feita nas linhas 1 até 12. A contração iterativa da rede

atual e a atualização de B são feitas até que o único vértice exposto de M respeito de B seja o vértice 1, de maneira que B resulta ser uma MAG de M.

A segunda fase, **Fase de Expansão** é feita nas linhas 13 até 18. Nesta fase, o lema último é aplicado iterativamente para obter uma MAG A de R_{YZ} a partir de B.

As estruturas C e β contém as informações necessárias para expansão recolhidas na fase de contração. Detalhes de ambas estruturas podem ser obtidos de [7].

A complexidade em tempo da fase de contração é $O(mlogn)$ e da fase expansão é $O(n)$, de maneira tal que o procedimento melhor pode ser implementado em tempo $O(mlogn)$.

O algoritmo das k melhores arborescências geradoras precisa de uma subrotina para encontrar uma seguinte melhor arborescência geradora (MAG) a partir de uma arborescência dada.

Seja $\overline{A}$ uma MAG da rede R sujeita as restrições $Y \subseteq \overline{A} \subseteq E - Z$. O problema consiste em encontrar uma seguinte melhor arborescência geradora sujeita às mesmas restrições.

Infelizmente, não é possível alcançar este objetivo tão facilmente como no caso análogo de árvores em grafos. Nesse caso, uma troca ótima de arestas é suficiente para encontrar uma seguinte melhor árvore.

Consideremos a rede $D = (V, E, w)$ com raiz 1 ilustrada na figura IV.1. A MAG de R, que é $\overline{A}$, 'e mostrada na figura IV.2 e a SAG é mostrada na figura IV.3, tendo esta última dois arcos distintos da MAG. Portanto, em geral, a idéia de troca ótima de arcos não serve.

Existe um método direto de encontrar uma SAG dado por Lawler [28]. Indexamos em qualquer ordem $e^{(1)}, e^{(2)}, \ldots, e^{(n-1)}$ os arcos de $\overline{A}$ e para cada arco $e^{(j)} \in \overline{A} - Y$ calculamos A_j (se existe), onde

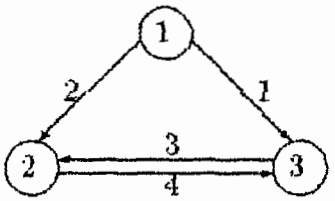


Figura IV.1: Dígrafo Original.

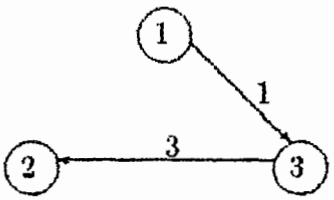


Figura IV.2: MAG.

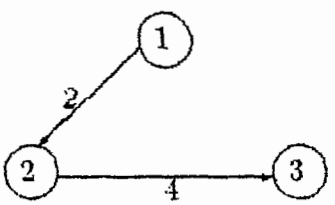


Figura IV.3: SAG.

$$A_j \text{ é uma MAG de } R_{Y_j Z_j} \tag{IV.1}$$

$$Y_j = Y \cup \{e^{(h)} \mid h < j\} \tag{IV.2}$$

$$Z_j = Z \cup \{e^{(j)}\} \tag{IV.3}$$

Assim, uma MAG A_j de peso mínimo necessita de $O(n)$ cálculos de MAG.

A importância do seguinte trabalho está na maneira eficiente de calcular uma SAG necessitando somente de um cálculo de MAG. Isto é possível devido à identificação eficiente de um arco pertencendo a MAG mas não pertencendo a SAG.

Para cada $b \in \overline{A} - Y$, sejam σ_M, τ_M, w_M respectivamente, as funções de incidência e peso da rede atual M consideradas em **melhor(Y,Z)** quando B há sido escolhido na linha 5.

Um arco f de M é chamado de **arco seguinte** a B se ele satisfaz:

$$\tau_M(f) = \tau_M(b);\ f \neq b \tag{IV.4}$$

$$\tau(b) \text{ não é predecessor de } \sigma(f) \text{ em } \overline{A} \tag{IV.5}$$

$$w_M(f) \text{ é mínimo entre todos os arcos de } M \text{ que satisfazem } IV.4 \text{ e } IV.5 \tag{IV.6}$$

No caso que tal arco exista, seja

$\delta(b) = w_M(b) - w_M(f)$; caso contrário $\delta = +\infty$.

O teorema seguinte dá uma propriedade importante relativa aos arcos seguintes a uma MAG definida em (IV.1), (IV.2) e (IV.3).

Teorema IV.1. Sempre é possível indexar os arcos de $\overline{A}$ de maneira tal que, para cada $e^{(j)} \in \overline{A} - Y$, ou existe A_j e $w(\overline{A}) - w(A_j) = \delta(e^{(j)})$, ou nenhum arco seguinte a $e^{(j)}$ existe e assim $\delta(e^{(j)}) = +\infty$.

Seja agora : $d = min_{b \in \overline{A}-Y} \delta(b)$, e se $d < +\infty$ seja $e \in \overline{A} - Y$ tal que $\delta(e) = d$.

Temos um corolário importante para o cálculo de uma SAG de $R_{Y\cdot Z}$.

Corolário IV.1. Se $R_{Y\cdot Z}$ não tem SAG então $d = +\infty$, senão:

(a) $A' =$ **melhor(Y, Z + e)** é uma SAG de de $R_{Y\cdot Z}$,

(b) $w(\overline{A}) - w(A') = d$.

O corolário IV.1 implica o seguinte método para calcular uma SAG A', **melhor(Y,Z)** é executado solucionando empates na linha 5 em favor dos arcos de $\overline{A}$, quando seja possível. Assim, sendo A uma MAG de $R_{Y\cdot Z}$ todos os arcos de $\overline{A}$ devem ser eventualmente escolhidos na linha 5. Cada vez que um arco $b \in \overline{A} - Y$ é escolhido, um arco f seguinte a b é procurado e $\delta(b)$ é obtido.

Portanto, podemos calcular d e e. Pelo corolário IV.1 se $d < +\infty$, $w(\overline{A}) - d$ é o peso de A' e **melhor(Y,Z+e)** dá uma SAG A' de $R_{Y\cdot Z}$. Se $d = +\infty$ então A' não existe.

O procedimento **seguinte** implementa as idéias anteriores. O procedimento auxiliar **procura** busca e retorna (se existe), um arco seguinte a b. Se b não tem arcos seguintes então **procura** retorna um arco artificial α de peso $w_M(\alpha) = -\infty$. Depois do término de **seguinte** as variáveis e, d identificam no sentido mencionado uma SAG de $R_{Y\cdot Z}$.

Procedure Procura(b)

Begin

Seja Q a fila usada para procurar b na linha 5 de **seguinte**;

While Q não está vazia **Do Begin**

$f \leftarrow min(Q)$;

If $\tau(b)$ não é um predecessor de

$\sigma(f)$ em A **Then Begin**

insira f em Q: **Return** f

End

End

$w_M(\alpha) \leftarrow -\infty$; **Return** α

End

Procedure seguinte(A,Y,Z):

Begin

$M \leftarrow R_{YZ}$; $B \leftarrow \emptyset$

$d \leftarrow +\infty$;

While exista um vértice exposto $v \neq 1$ de M respeito de B **Do Begin**

Sejam σ_M, τM. w_M respectivamente as funções de incidência e peso de M;

Seja b um arco de M tal que $\tau(b) = v$ e w_M é mínimo;

Comment quando for possível, os empates em 5 devem ser resolvidos em favor dos arcos de A;

$B \leftarrow B + b$;

If $b \in A - Y$ **Then Begin**

$f \leftarrow$**procura**(b);

If $w_M(b) - w_M(f) < d$ **Then Begin**

$e \leftarrow b$; $d \leftarrow w_M(b) - w_M(f)$

End

End

If B contém um circuito C de M **Then Begin**

$M \leftarrow M_c$; $B \leftarrow B - C$

End

End

End

Observamos que o procedimento **seguinte** é similar a **melhor**, sendo as linhas 8 até 11 distintas. Como agora A é conhecida, a atualização de C, β e a fase de expansão não são necessárias. Assim, **seguinte** pode ser eficientemente implementado utilizando as mesmas estruturas de dados ocupadas para **melhor**. Tais estruturas podem ser encontradas no artigo de Tarjan [37]. A complexidade de **seguinte** é $O(mlogn)$.

Agora estamos em condições de descrever o algoritmo para encontrar as k MAG na ordem não decrescente de pesos de uma rede.

Lawler [28] dá um esquema geral de ranking para problemas de Otimização Combinatória. Implementaremos esse esquema seguindo o artigo de Gabow [21].

Sejam $A^{(i)}$, para $1 \leq i \leq k$, as k MAG de R na ordem não decrescente de pesos. Suponhamos que o algoritmo gerou $A^{(1)}$, $A^{(2)}$, ..., $A^{(j-1)}$, $j > 1$. As AG restantes são então particionadas em $j-1$ conjuntos disjuntos da forma:

$$P_i^{(j-1)} = \{A^{(h)} \mid h > j-1, Y_i^{(j-1)} \subseteq A^{(h)} \subseteq E - Z_i^{(j-1)}\},\ 1 \leq i \leq j-1.$$

Claramente $A^{(i)}$ satisfaz todas as condições de $P_i^{(j-1)}$ exceto $i > j-1$. Logo, uma MAG em $P_i^{(j-1)}$ é identificada por uma SAG A' tal que:

$$Y_i^{(j-1)} \subseteq A' \subseteq E - Z_i^{(j-1)}.$$

O algoritmo usa uma lista P para representar a partição. Cada conjunto $P_i^{(j-1)}$ é representado por uma tupla (w, e, A, Y, Z) em P, onde $A = A^{(i)}$ é uma MAG de R sujeita às restrições:

$$Y_i^{(j-1)} \subseteq A \subseteq E - Z_i^{(j-1)};$$

w é o peso de uma SAG A' de R_{YZ}. O arco e identifica A' no sentido que uma SAG A' de R_{YZ} pode ser obtida por **melhor(y, Z + e)**. Assim, o passo geral do algoritmo de ranking é o seguinte:

Identificar a tupla para a qual w é mínimo, executar **melhor(Y, Z + e)** para obter $A^{(j)}$; formar uma nova partição $P_h^{(j)}, 1 \leq h \leq j$, subdividindo $P_i^{(j-1)}$, o conjunto representado pela tupla escolhida. $P_i^{(j-1)} - \{A^{(j)}\}$ é particionado em dois subconjuntos: um composto de arborescências contendo o arco e, e outro composto de arborescências não contendo e. Duas execuções do procedimento **seguinte** podem fazer o trabalho necessário, **seguinte**$(A^{(j)}, Y + e, Z)$ e **seguinte**$(A^{(j)}, Y, Z + e)$. Escrevamos agora o algoritmo de ranking.

Procedure Ranking(R,k)
Begin
$A^{(1)}$ —**melhor**$(\emptyset, \emptyset)$: **Output** $A^{(1)}$;
seguinte$(A^{(1)},\emptyset,\emptyset)$; faça $(w(A^{(1)}) - d, e, A^{(1)}, \emptyset, \emptyset)$ a única tupla em P;
For $j \leftarrow 2$ **Until** k **Do Begin**
elimine de P uma tupla (c, e, A, Y, Z) tal que w é mínimo;
Comment esta tupla representa $P_i^{(j-1)}$;
If $w = -\infty$ **Then**
Return 'acabaram as AG de R';
$Y' \leftarrow Y + e$; $Z' \leftarrow Z + e$;
$A^{(j)}$ —**melhor**(Y, Z'); **Output** $A^{(j)}$;
seguinte(**A**,**Y'**,**Z**); adicione $(w(A) - d, e, A, Y', Z)$ a P;
seguinte$(A^{(j)}, Y, Z')$; adicione $(w(A^{(j)}) - d, e, A^{(j)}, Y, Z')$ a P;
Comment note que **seguinte** atualiza variáveis globais e, d.
End
End

Façamos um breve estudo da complexidade do algoritmo.

Cada iteração da linha 4 a 11 necessita tempo $O(mlogn)$, mais o tempo necessário para manipular P, onde $O(mlogn)$ é a complexidade para **melhor e seguinte**. O tempo para manipular P é $O(m)$ já que P pode ser memorizada usando uma estrutura heap. O heap contém no máximo k elementos e assim um elemento pode ser eliminado e adicionado em tempo $O(logk)$. Como $k < 2^m$, $O(m)$ é o tempo necessário para manipular P e logo o tempo total é $O(kmlogn)$ para **ranking**.

IV.3 Formulação do Problema como um Problema de Arborescência Geradora Restrita

Seja $D = (V, E, w)$ uma rede com raiz $r = 1$. Para formular o PSD como um problema de arborescência geradora restrita precisamos construir a partir da rede D uma nova rede $D' = (V', E', w')$ que chamaremos de **rede aumentada** e que é construída:

1. Adicionando um vértice artificial $s = 0$ a V.
2. Para cada vértice $i \in V_f$ adicionamos um arco (s, i) com peso associado $w_{si} = 0$.
3. Para a raiz r adicionamos um arco (s, r) com peso $w_{sr} = 0$.

Então $V' = V + s$ e $E' = E \cup \{(s, i) \mid i \in V_f\} \cup \{(s, r)\}$.

Como $I(r) = \emptyset$ em D, (s, r) pertence a qualquer r-arborescência geradora de D'. Se eliminamos (s, r) em tal arborescência obteremos duas componentes conexas de D'; elas são arborescências enraizadas em r e s.

Suponhamos que T_p é uma r-arborescência geradora de D' e T_{pr} e T_{ps} são as componentes enraizadas em s e r, respectivamente. Quando em $T_{pr} = (V_{pr}, E_{pr})$ temos $V_0 \subseteq V_{pr}$ então $w(T_p) = w(T_{pr})$ é um limite superior de $w(T^*)$, isto é, $w(T^*) \leq w(T_{pr})$.

Seja T_1 a r-arborescência geradora ótima de D'. É simples observar que $w(T_1) \leq w(T^*)$. Se aplicamos o algoritmo de ranking, já estudado na seção anterior, de Camerini, Fratta e Maffioli [7], para calcular as k melhores s-arborescências geradoras de D' obteremos:

$$w(T_p) \leq w(T_{p+1}), p = 1, 2, \ldots, k-1.$$

Suponhamos que T_q, $1 \leq q \leq k$, é a primeira r-arborescência geradora de D' (dada pelo algoritmo de ranking) para a qual T_{qr} é viável, isto é, em $T_{qr} = (V_{qr}, E_{qr})$ temos que $V_0 \subseteq V_{qr}$, então T_{qr} é uma r-arborescência de Steiner ($T_{qr} = T^*$) de D.

Ilustramos este algoritmo utilizando o dígrafo inicial da figura IV.4. Nas figuras IV.5–IV.12 podemos temos as 7 melhores r-arborescências geradoras de D sendo que na figura IV.12 temos T^*.

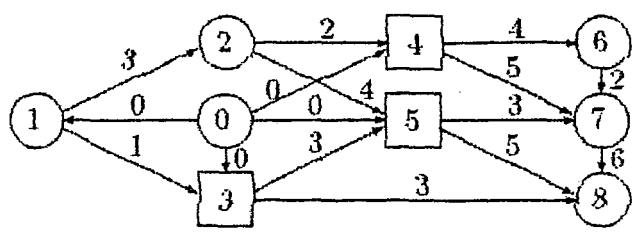


Figura IV.4: Dígrafo Aumentado

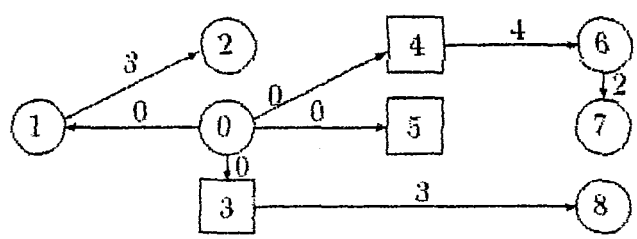


Figura IV.5: $w(T_1) = 12$.

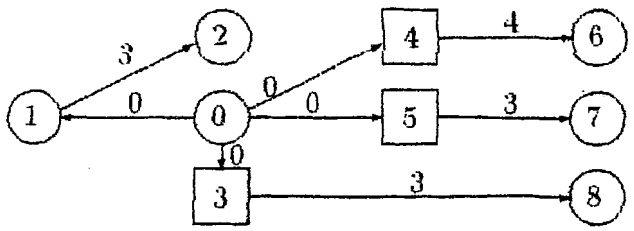


Figura IV.6: $w(T_2) = 13$.

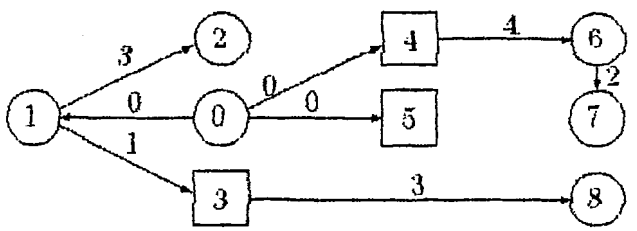


Figura IV.7: $w(T_3) = 13$.

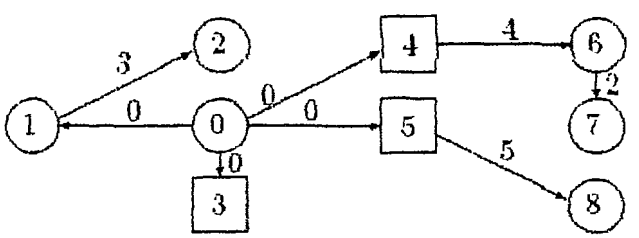


Figura IV.8: $w(T_4) = 14$.

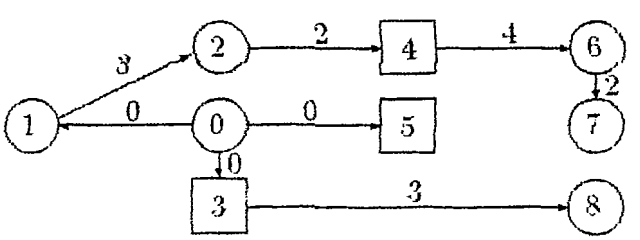


Figura IV.9: $w(T_5) = 14$.

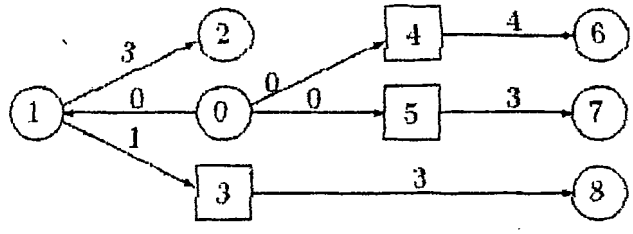


Figura IV.10: $w(T_6) = 14$.

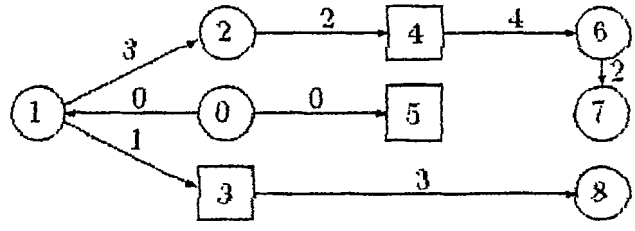


Figura IV.11: $w(T_7) = 15$.

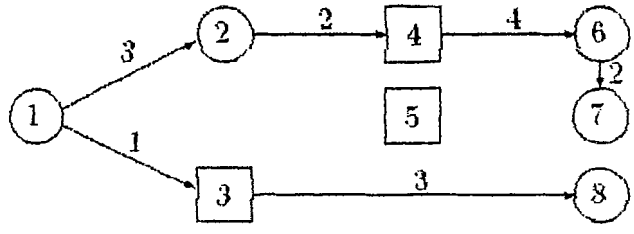


Figura IV.12: Solução Ótima. $w(T^*) = w(T_7) = 15$.

IV.4 Resultados Computacionais e Conclusões

Daremos aqui os resultados numéricos obtidos na implementação computacional da formulação do PSD como um problema de arborescência geradora restrita.

O esquema de trabalho foi o seguinte. Primeiro, foram criadas séries de instâncias geradas aleatoriamente, na mesma forma que fizemos com os métodos de redução. Depois, aplicamos repetidamente os testes de redução até que nenhuma redução adicional fosse obtida; os pesos estão no intervalo [1,10]. Finalmente o algoritmo de ranking foi aplicado ao dígrafo aumentado (construído depois do processo de redução) listando arborescências geradoras até encontrar a solução ótima.

Usamos uma série de 10 problemas gerados aleatoriamente para cada configuração de $|V|$, $|V_f|$ e $|E|$. A tabela IV.1 dá um resumo da experiência computacional; os resultados são médias sobre os 10 problemas arredondados a valores inteiros. A última coluna indica o desvio padrão associado ao número de arborescências geradoras de D' listadas.

Valores Iniciais			Depois das Reduções			Número de Arborescências Geradoras de G' Listadas	Desvio
$\|V\|$	$\|V_f\|$	$\|E\|$	$\|V\|$	$\|V_f\|$	$\|E\|$		
20	10	50	8	2	17	244	343
30	15	60	11	4	20	1785	1742
40	20	70	9	2	17	882	1255

Tabela IV.1: Resumo da experiência computacional com os métodos de redução e algoritmo de ranking.

A perfomance do algoritmo de ranking deveria melhorar pela inclusão de limites superiores e testes de redução em passos intermediários do algoritmo eliminando arborescências geradoras improdutivas de D'.

Este algoritmo ascendente poderia ser útil quando estivermos interessados em encontrar todas as soluções ótimas para um problema.

Depois da obtenção de uma solução ótima com peso k, o algoritmo de ranking pode entregar todas as outras arborescências com o mesmo peso. Qualquer outra arborescência ótima deve estar neste conjunto de arborescências com peso k.

Finalmente, mencionamos uma propriedade interessante do algoritmo de ranking. Em cada passo do algoritmo ele dá uma solução quase viável e assim poderia servir de base para um procedimento heurístico. Lembramos que, em cada passo, o algoritmo identifica o melhor sucessor em cada partição. A parte T_{pr} de cada sucessor T_p é uma solução parcial para o PSD. Nesta direção foi sugerido um esquema que combina o algoritmo de ranking para encontrar limites inferiores e a heurística de Palma–Pacheco [33] para encontrar limites superiores, ver [31] para detalhes.æ

Capítulo V

Conclusões

O Problema de Steiner em Dígrafos é um problema de Otimização Combinatória que tem interessantes aplicações na área de Desenho Ótimo de Redes.

Para este problema construímos um conjunto de testes de redução, os quais foram implementados em tempo polinomial, e que reduzem o espaço de soluções para o problema fixando variáveis. No nosso caso, a fixação de variáveis significa principalmente a eliminação de arcos do dígrafo que não podem pertencer à solução ótima.

A experiência computacional mostrou que o conjunto de testes permite diminuir fortemente o tamanho do dígrafo ainda no caso de redes esparsas. Sugerimos criar novos testes que sejam capazes de detetar vértices facultativos que necessariamente devam participar da solução ótima e também aqueles que não possam participar da solução ótima.

Também implementamos um algoritmo exato de enumeração de arborescências para o problema. Para instâncias aleatoriamente geradas, aplicamos primeiro os testes de redução e depois aplicamos o algoritmo de ranking para encontrar a solução ótima.

Os resultados obtidos foram bons no sentido que o algoritmo é aplicável a instâncias maiores e a perfomance do algoritmo pode ser ainda melhorada.

O algoritmo parece particularmente útil para redes esparsas e even-

tualmente poderia ser ocupado para gerar boas soluções heurísticas.

Finalmente, notamos o fato que o algoritmo de enumeração poderia também ser usado para solucionar outros problemas de arborescências geradoras mínimas restritas, ainda no caso que estas restrições não estejam representadas numa forma matemática explícita

Referências Bibliográficas

[1] U. Akinc e B. Khumawala. An Efficient Branch and Bound Algorithm for the Capacited Warehouse Location Problem. *Management Science*, 23:585–594, 1977.

[2] A. Balakrishnan e N. Patel. Problem Reduction Methods and a Tree Generation Algorithm for the Steiner Network Problem. *Networks*, 17:65–85, 1987.

[3] M. O. Ball, W. Liu e W. R. Pulleyblank. *Two Terminal Steiner Tree Polyhedra*. Technical Report 87021, Management Science and Statistics, University of Maryland at College Park, 1987.

[4] J. E. Beasley. An Algorithm for the Steiner Problem in Graphs. *Networks*, 14:147–159, 1984.

[5] J. E. Beasley. An SST-Based Algorithm for the Steiner Problem in Graphs. *Networks*, 19:1–16, 1989.

[6] J. F. Benders. Partitioning Procedures for Solving Mixed Variables Programming Problems. *Numerische Mathematik*, 4:238–252, 1962.

[7] P. Camerini, L. Fratta e F. Maffioli. The k Best Spanning Arborescences of a Network. *Networks*, 10:91–110, 1980.

[8] A. E. Candia. *Problemas de Arborescências em Grafos*. Tese Mestrado, COPPE/Sistemas-UFRJ, Brasil, 1988.

[9] S. Chopra e M. R. Rao. *The Steiner Tree Problem I: Formulation, Composition and Extension of Facets*. Technical Report, University of New York, 1989.

[10] J. Choquette, M. Dror e B. Gavish. Directed Steiner Tree Problem on a Graph: Models, Relaxations and Algorithms. *INFOR*, 3:266–281, 1990.

[11] A. Claus e N. Maculan. *Une Nouvelle Formulation du Problème de Steiner sur un Graphe*. Relatório Técnico, Centre de Recherche sur les Transports, 1983.

[12] E. W. Dijkstra. A Note on Two Problems in Connection with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[13] C. W. Duin e A. Volgenant. Reduction Tests for the Steiner Problem in Graphs. *Networks*, 19:549–567, 1989.

[14] C. W. Duin e A. Volgenant. An Edge Elimination Test for the Steiner Problem in Graphs. *Operations Research Letters*, 8:79–83, 1989.

[15] C. W. Duin e A. Volgenant. Some Generalizations of the Steiner Problem in Graphs. *Networks*, 17:353–364, 1987.

[16] J. Edmonds. Optimum Branching. *Journal of Research of the National Bureau of Standards*, 71B-4:233–240, 1967.

[17] S. Even. *Graph Algorithms*. Computer Science Press, Maryland, 1979.

[18] C. E. Ferreira. *O Problema de Steiner em Grafos: Uma Abordagem Poliédrica*. Tese Mestrado, Universidade de São Paulo, 1989.

[19] R. W. Floyd. Algorithm 97: Shortest Path. *CACM* 5:345–345, 1962.

[20] L. Ford e D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[21] H. N. Gabow. Two Algorithms for Generating Weighted Spanning Trees in Order. *SIAM J. Computing*, 6:139–150, 1977.

[22] M. Gondran e M. Minoux. *Graphs and Algorithms*. J.Wiley & Sons, 1984.

[23] L. Guyard. *Le Problème de l'Arbre de Steiner: Modelisation par Programmation Linéaire et Resolution par des Techniques de Décomposition (Application à un Modèle de Donées Relationelles*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, 1985.

[24] S. L. Hakimi. Steiner's Problem in Graphs and its Implications. *Networks*, 1:113–133, 1971.

[25] M. Hanan. On Steiner's Problem with Rectilinear Distance. *SIAM J.Appl.Math.*, 14:255–265, 1966.

[26] G. P. Ingargiola e J. F. Korsh. Reduction Algorithm for Zero–One Single Knapsack Problems. *Management Science*, 20:460–463, 1973.

[27] R. M. Karp. *Reducibility Among Combinatorial Problems.* Plenum Press, 1972.

[28] E. L. Lawler. A Procedure for Computing the k Best Solutions to Discrete Optimization Problems and its Applications to the Shortest Path Problems. *Management Science*, 18:401–405, 1972.

[29] W. Liu. A Lower Bound for the Directed Steiner Tree Problem in Directed Graphs. *Networks*, 20:765–778, 1990.

[30] N. Maculan. The Steiner Problem in Graphs. *Annals of Discrete Mathematics*, 31:185–212, 1987.

[31] N. Maculan, P. M. Souza e A. Candia. An Approach for the Steiner Problem in Directed Graphs. A ser publicado.

[32] S. Nguyen, D. Arpin, N. Maculan. *Le Probléme de Steiner sur un Graphe Orienté: Formulations et Relaxations.* Relatório Técnico, Centre de Recherche sur les Transports, Université de Montréal, Canada, 1983.

[33] O. L. Palma-Pacheco. *Contribuição para a Resolução do Problema de Steiner num Grafo Direcionado: um Método Heurístico.* Tese de Doutorado, COPPE/Sistemas-UFRJ, Brasil, 1985.

[34] S. M. Selkow, L. Nastansky e N. F. Stewart. Cost–Minimal Trees in Directed Acyclic Graphs. *Zeitschrift fur Operations Research*, 18:59–67, 1974.

[35] C. C. de Souza. *O Problema de Steiner na Métrica Retilínear: Propriedades, Novas Heurísticas e Estudo Computacional.* Tese Mestrado, PUC-Rio de Janeiro, 1989.

[36] J. Szwarcfiter. *Grafos e Algoritmos Computacionais.* Editora Campus, 1984.

[37] R. E. Tarjan. Finding Optimum Branchings. *Networks*, 7:25–35, 1977.

[38] S. Voss. A Reduction Based Algorithm for the Steiner Problem in Graphs. *Methods of Operations Research*, 58:239–252, 1989.

[39] P. Winter. An Algorithm for the Steiner Problem in the Euclidean Plane. *Networks*, 15:323–345, 1985.

[40] P. Winter. Steiner Problem in Networks – a Survey. *Networks*, 17:129–167, 1987.

[41] P. Winter e J. MacGregor Smith. *Heuristics for the Steiner Problem in Networks.* Technical Report, University of Massachussets, 1990.

[42] R. T. Wong. A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph. *Mathematical Programming*, 28:271–287, 1984.