O problema da designação e sua variante paramétrica

Lidio Nunes de Abreu Junior

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

O problema da designação e sua variante paramétrica

Lidio Nunes de Abreu Junior[1]

maio de 2000

Banca Examinadora:

- Prof. Dr. João Carlos Setubal
  IC/UNICAMP (Presidente)
- Prof. Dr. Carlos Eduardo Ferreira
  IME/USP
- Prof. Dr. Cid Carvalho de Sousa
  IC/UNICAMP
- Prof. Dr. João Meidanis (Suplente)
  IC/UNICAMP

[1]Supported in part by CNPq grant 131954/1997-0. FAPESP grant 97/11222-0

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Abreu Junior, Lídio Nunes de

Ab86p O problema da designação e a sua variante paramétrica / Lídio Nunes de Abreu Junior -- Campinas, [S.P. :s.n.], 2000.

Orientador : João Carlos Setubal

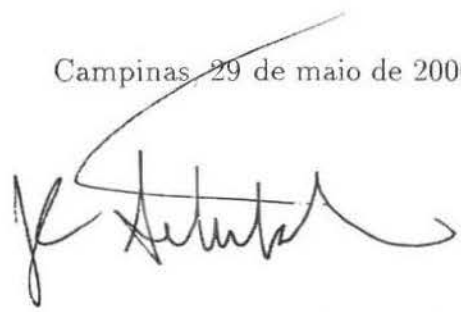
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. I. Setubal, João Carlos. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

O problema da designação e sua variante paramétrica

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Lidio Nunes de Abreu Junior e aprovada pela Banca Examinadora.

Campinas, 29 de maio de 2000.

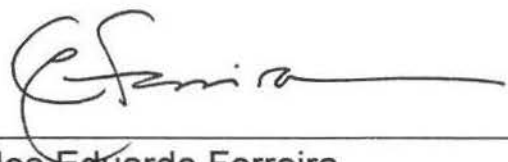


Prof. Dr. João Carlos Setubal
IC/UNICAMP (Presidente)

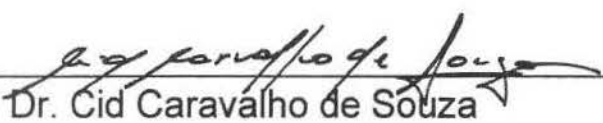
Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 29 de maio de 2000, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. Carlos Eduardo Ferreira
IME - USP



Prof. Dr. Cid Caravalho de Souza
IC-UNICAMP

Prof. Dr. João Carlos Setubal
IC-UNICAMP

"É melhor tentar e falhar que preocupar-se e ver a vida passar. É melhor tentar, ainda que em vão, que sentar-se fazendo nada até o final. Eu prefiro na chuva caminhar, que em dias tristes em casa me esconder. Prefiro ser feliz, embora louco, que em conformidade viver"

Martin Luther King

Dedicatória

Aos meus pais.

Agradecimentos

A Valdemi de Bastos pelo carinho, preocupação e pelas palavras de incentivo em todas as etapas deste estudo, por entender e apoiar os meus ideais.

A Adriana Dias por toda a sua colaboração, incentivo e amizade.

Ao professor João Carlos Setubal pelo profissionalismo, competência e dedicação constantes imprescindíveis para a realização deste trabalho. Agradeço sobretudo as suas críticas, a paciência e a amizade demonstrada.

A Andrew Goldberg pela disponibilização de código de grande ajuda neste trabalho.

Aos amigos que encontrei nesta cidade e que passam por situações semelhantes. A Pascual, Guilherme Albuquerque, Guilherme Pimentel, José Roberto, Héctor, André e Cesar agradeço o companheirismo e amizade.

A todos estes, e aos demais, que direta ou indiretamente me apoiaram na concretização deste trabalho.

Resumo

O assunto principal desta tese é o problema da designação: dado um grafo bipartido com custos nas arestas, obter um emparelhamento perfeito de custo mínimo. Na primeira parte do trabalho apresentamos uma revisão detalhada dos conceitos e principais resultados da literatura sobre esse problema. Na segunda parte, discutimos uma variante paramétrica, na qual cada aresta e tem seu custo dado por uma expressão do tipo $c_e^\circ + \lambda c_e^\bullet$, onde c_e° e $c_e^\bullet$ são constantes e λ é o parâmetro, cujo valor é real e varia. Nesta variante o objetivo é obter todas as soluções do problema para um intervalo de valores de λ no menor tempo possível. Nesta parte inicialmente fazemos uma revisão de resultados da literatura que apresentam técnicas gerais para a resolução de problemas paramétricos em Otimização Combinatória. Em seguida apresentamos uma abordagem, também da literatura, específica para o problema do fluxo de custo mínimo, do qual o problema da designação é um caso especial. Esta abordagem se baseia em propriedades do algoritmo simplex de rede. Em seguida apresentamos uma nova abordagem, baseada numa relação entre o problema do fluxo de custo mínimo e o problema do ciclo de razão mínima. A complexidade desta nova abordagem é insatisfatória quando se quer resolver uma instância do problema da designação paramétrico, pois a complexidade conhecida do problema do ciclo de razão mínima é maior do que a complexidade conhecida do problema da designação. Esta nova abordagem entretanto é satisfatória do ponto de vista teórico quando aplicada ao problema do fluxo de custo mínimo paramétrico. O trabalho finaliza apresentando uma comparação experimental entre as abordagens "simplex de rede" e "ciclo de razão mínima" para o problema do fluxo mínimo paramétrico, mostrando que a primeira é muito superior à segunda.

Abstract

This dissertation is about the Assignment Problem: given an edge-weighted bipartite graph, find a perfect matching of minimum weight. In the first part of this work we present a detailed survey of the literature on this problem, including basic concepts and main results. In the second part, we discuss a parametric variant of this problem. In this variant, the weight of each edge e is given by $c_e^\circ + \lambda c_e^\bullet$, where c_e° and $c_e^\bullet$ are constants and λ is the parameter, that is, a real value that can vary. For a given range of λ values we want to find all solutions to the corresponding assignment problems in the least possible time. In this part of the work we initially present a survey of techniques for solving general Combinatorial Optimization parametric problems. We then present a technique, also from the literature, for solving the parametric minimum cost flow problem, of which the assignment problem is a special case. This technique is based on the network simplex algorithm. We then present a new technique, also for solving the parametric minumum cost flow problem, based on a reduction to the minimum cost-to-time ratio cycle problem. This technique is not satisfactory for solving parametric assignment problems, because the best known algorithm for the minimum cost-to-time ratio cycle problem has worst running time than the best algorithm known for the assignment problem. It is however satisfactory from a theoretical point of view when applied to parametric minimum cost flow problems, because its running time is better than the best known running time for a certain class of inputs. We made an experimental comparison between the "network simplex" approach and the "minimum ratio cycle approach", but found that in all cases the "network simplex" approach has faster running times.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Em diversas ocasiões temos a necessidade de associar aos pares os elementos de um conjunto V da melhor maneira possível. Para caracterizar uma determinada associação como sendo melhor em relação a outra, pressupõe-se que de alguma maneira pode-se estipular o benefício ou custo envolvido em associar um par de elementos $x, y \in V$.

Associações desta natureza são uma importante classe de problemas em Otimização Combinatória denominada **problemas de emparelhamentos**. Dentre as diversas variantes existentes de emparelhamento, algumas se destacam por receberem atenção especial. Uma variante é o problema de *emparelhamento bipartido*. Nesta circunstância o conjunto V pode ser particionado em subconjuntos X e Y. O objetivo é emparelhar da melhor maneira os elementos do subconjunto X aos elementos do subconjunto Y.

O modelo fundamental estudado neste trabalho é o *problema de emparelhamento bipartido de custo mínimo*, também denominado o *problema da designação*[1]. O problema da designação pode aparecer em muitas situações práticas. Os subconjuntos X e Y podem ter vários significados dependendo do contexto em que se inserem. Por exemplo, o conjunto X pode representar funcionários e o conjunto Y tarefas a serem realizadas, e neste contexto o objetivo é *designar* um funcionário para a realização de cada tarefa. Em outro contexto pode-se ter a necessidade de associar, por exemplo, caminhões a rotas de viagem.

1.1 Grafos e emparelhamentos

Nesta seção formalizamos os problemas de emparelhamento e na seção 1.3 o problema principal deste estudo. Introduzimos a seguir algumas notações e definições básicas utilizadas neste texto. Os termos apresentados são suficientes apenas para o entendimento do problema no seu contexto em Teoria dos Grafos. Para a explicação dos resultados

[1] A denominação em inglês para este problema é *assignment problem*

mais relevantes e para uma análise do problema, conceitos e definições adicionais serão expostos no capítulo 2.

A visão informal apresentada anteriormente sobre o problema de emparelhamento teve o objetivo de introduzir de maneira simples a essência do problema. Para discutirmos todos os aspectos teóricos e computacionais relacionados necessitamos de um modelo formal. O modelo adequado às nossas necessidades é fornecido pela Teoria dos Grafos[2].

Uma maneira boa para modelar uma associação envolvendo os elementos de um conjunto é aplicar o conceito de grafos. A *Teoria dos Grafos* é um modelo formal capaz de abstrair as mais variadas situações envolvendo elementos e seus relacionamentos.

Formalmente, um grafo $G = (V, E)$ é um conjunto de objetos V denominados *vértices* e um conjunto de *arestas* E modelando relações entre pares de vértices. A cardinalidade de V será denotada por $|V| = n$ e a cardinalidade de E por $|E| = m$. Neste texto nos referiremos a uma aresta $e = (v, w) \in E$ pelo par de vértices (v, w) incidente à mesma ou pelo seu identificador e.

Pode-se definir grafos ponderados ou não. Se o grafo é ponderado para cada aresta $e \in E$ está associado um valor c_e (custo, benefício, etc) para quantificar a relação entre o par de vértices incidente à mesma.

As arestas de um grafo podem ser orientadas ou não. Se as arestas são *orientadas* a ordem dos vértices é importante. As arestas orientadas são pares ordenados e nestas circunstâncias as arestas (v, w) e (w, v) são distintas. Um grafo G é *orientado* se as suas arestas são orientadas, caso contrário, o grafo é *não orientado*.

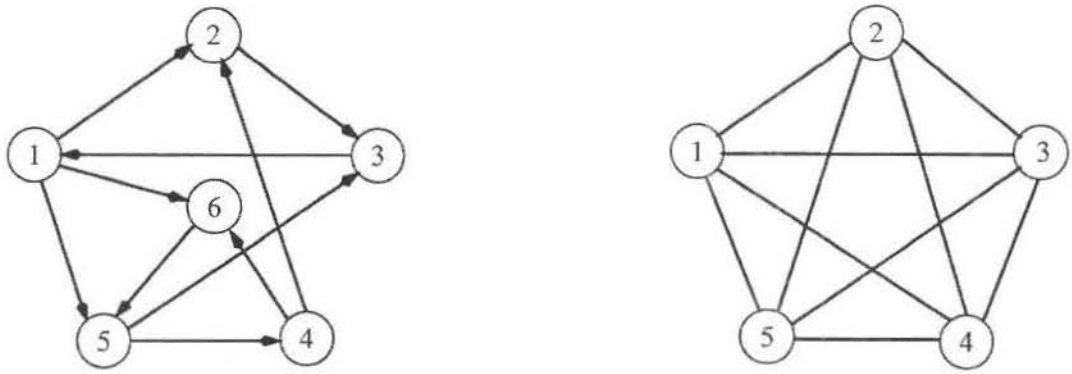
Pode-se definir um *passeio* como sendo uma sequência de vértices $v_1, v_2, \ldots, v_k$ tais que os mesmos são conectados por arestas $(v_1, v_2), \ldots, (v_{k-1}, v_k)$. Um *caminho* é um passeio no qual nenhum vértice está presente mais de uma vez, com a possível exceção do primeiro e último vértices. Passeios e caminhos são definidos tanto em relação a grafos orientados quanto a grafos não orientados.

Um *ciclo* consiste em um caminho no qual o primeiro e o último vértice são idênticos. A definição de ciclo também se aplica aos grafos orientados e não orientados.

Um grafo G é *completo* se para todo par de vértices $v, w \in V$ a aresta $(v, w) \in E$ (veja figura 1.1). Um grafo G é *bipartido* se o conjunto de vértices V pode ser particionado em dois subconjuntos X e Y de forma que toda aresta conecta um vértice de X a um vértice de Y. Os subconjuntos X e Y são as *partições* de V.

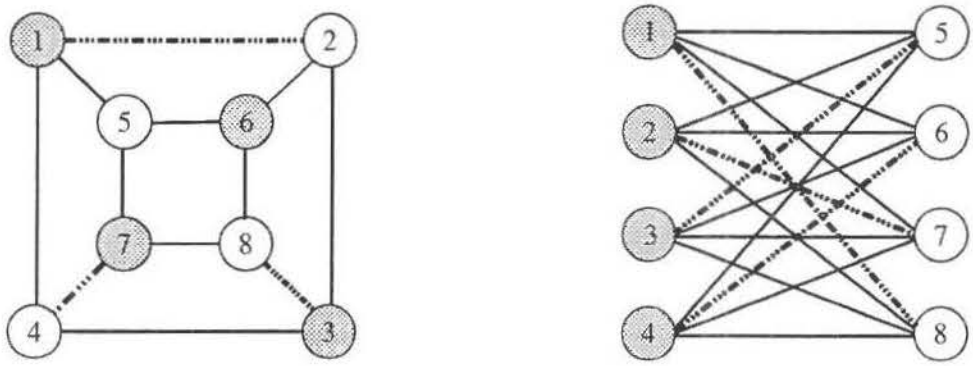
Um **emparelhamento** $\mathcal{M}$ em um grafo G é um subconjunto $\mathcal{M} \subseteq E$ de arestas tal que duas arestas quaisquer de $\mathcal{M}$ não possuam um vértice em comum. As arestas presentes no emparelhamento $\mathcal{M}$ são denominadas *emparelhadas*. As demais arestas são denominadas *não emparelhadas*. Da mesma maneira, os vértices incidentes às arestas de

[2]Gondran e Minoux [GM84] e West [Wes96] fornecem um estudo detalhado sobre a Teoria dos Grafos e os aspectos computacionais envolvidos



grafo orientado *grafo completo*

Figura 1.1: *Exemplos de grafos*



emparelhamento parcial *emparelhamento perfeito*

Figura 1.2: *Grafos bipartidos*

$\mathcal{M}$ são denominados *emparelhados*. Os demais vértices não incidentes à nenhuma aresta de $\mathcal{M}$ são denominados *não emparelhados*.

Um emparelhamento $\mathcal{M}$ é denominado *perfeito* se todos os vértices são vértices emparelhados, caso contrário, o emparelhamento é denominado *parcial*(veja figura 1.2).

Os problemas de emparelhamentos podem ser definidos em grafos ponderados ou não. Se custos são associados às arestas de G, o *custo* do emparelhamento $\mathcal{M}$ consiste na soma dos custos das arestas emparelhadas, o qual denotaremos por:

$$c_{\mathcal{M}} = \sum_{e \in \mathcal{M}} c_e$$

Se capacidades mínima l_e e máxima u_e não negativas são associadas às arestas pode-se definir o conceito de fluxo. Um **fluxo** f em G é uma função das arestas a valores não negativos tal que as seguintes condições são satisfeitas:

$$l_{vw} \leq f_{vw} \leq u_{vw} \qquad \forall (v,w) \in E \qquad \textit{capacidade}$$
$$\sum_{w \in V} f_{vw} = \sum_{w \in V} f_{wv} \qquad \forall v \in V \qquad \textit{conservação}$$

Neste texto denotaremos o maior custo de uma aresta em G por C e a maior capacidade por U.

1.2 Complexidade computacional

A complexidade intrínseca de cada problema é difícil ser avaliada. Até conseguirmos expressar a dificuldade de um determinado problema e obtermos um algoritmo que resolva-o da maneira mais eficiente possível, estaremos empenhados em melhorar o desempenho dos algoritmos existentes para este problema.

A *eficiência* de um algoritmo depende do critério utilizado como medida de complexidade. Critérios possíveis são o tempo de execução, o espaço de memória, o número de processadores, etc. No nosso caso estamos interessados principalmente no *tempo de execução* dos algoritmos.

Descrever o comportamento exato dos algoritmos é muito difícil. Nos restringiremos a obter uma aproximação da sua complexidade que nos permita avaliá-lo em relação a outro algoritmo. O tempo de execução deve ser definido em função do número de operações relevantes para o problema independente de qualquer arquitetura de computador.

O tempo necessário para se resolver uma instância do problema depende do seu tamanho. Consideramos que o *tamanho* da instância é o número total de bits necessários para representar os dados da entrada em uma notação apropriada, como por exemplo, a notação binária.

Estamos interessados na análise de *pior-caso* da complexidade dos algoritmos. Expressaremos a **complexidade** dos algoritmos pelo maior tempo necessário para resolver cada possível instância do problema em função do seu tamanho. O tempo de execução assintótico será utilizado para descrever o comportamento do tempo de execução dos algoritmos quando o tamanho da instância cresce infinitamente. Nos problemas estudados neste trabalho os dados de entrada são normalmente os vértices, as arestas, os custos e capacidades das arestas do grafo. Desta maneira o tamanho da entrada do problema é da ordem de $O(n+m)$.

Consideramos os algoritmos **eficientes** ou de *tempo polinomial* quando sua complexidade pode ser limitada por um polinômio em função do tamanho da entrada. A denominação algoritmos de *tempo superpolinomial* é utilizada em algoritmos cuja complexidade não pode ser limitada por um polinômio em função do tamanho da entrada. Estes algoritmos têm comportamento aceitável apenas para instâncias de pequena magnitude.

Para instâncias suficientemente grandes os algoritmos de tempo superpolinomial tornam-se inviáveis.

Os algoritmos de *tempo pseudopolinomial* têm complexidade em função de algum parâmetro da entrada do problema além do seu tamanho. Um dos parâmetros da entrada que estará presente freqüentemente na complexidade dos algoritmos analisados é o maior custo C de uma aresta da instância, como por exemplo, $O(nC)$.

Os algoritmos de tempo pseudopolinomial correspondem à uma importante subclasse dos algoritmos de tempo superpolinomial. Sob determinadas condições os algoritmos de tempo pseudopolinomial possuem comportamento similar ao dos algoritmos de tempo polinomial. Uma dessas condições é a **hipótese de similaridade** na qual consideramos que os dados podem ser limitados polinomialmente. No caso dos problemas em grafos consideramos que $C = O((n+m)^k)$ para alguma constante k. A importância desta subclasse de algoritmos está relacionada ao fato de que a hipótese de similaridade é satisfeita por um grande número de problemas com aplicações práticas e teóricas.

Um inteiro $\mathcal{C}$ pode ser representado com $\log C$ bits. Algoritmos com complexidade $O(m \log C)$, $O(n^2 \log\log C)$ e $O(nm \log(nC))$ são portanto claramente algoritmos de tempo polinomial. Para diferenciar as classes de algoritmos de tempo polinomial utiliza-se a denotação algoritmos de tempo polinomial forte e fraco. Os algoritmos de *tempo polinomial forte* são os algoritmos com complexidade estritamente em função do tamanho da entrada. Os demais algoritmos de tempo polinomial, com complexidade em função de parâmetros da entrada além do tamanho, são denominados algoritmos de *tempo polinomial fraco*.

Os algoritmos de tempo polinomial fraco possuem uma desvantagem em relação aos algoritmos de tempo polinomial forte. O comportamento dos algoritmos de tempo polinomial fraco pode ser indesejável para grandes valores de $\mathcal{C}$. Além disso, estes algoritmos podem ter problemas quando os dados de entrada são irracionais. Desta maneira, algoritmos de tempo polinomial forte são preferidos, sempre que possível[3].

1.3 O problema da designação

Seja $G = (V, E)$ um grafo bipartido ponderado não orientado com partições $V = X \bigcup Y$. O **problema da designação** consiste em determinar um emparelhamento perfeito $\mathcal{M}$ de custo mínimo em G. O problema da designação neste texto será abreviado por PD.

Sem perda de generalidade suporemos que a cardinalidade das partições é $|X| = |Y| = n$ ($|V| = 2n$). Suporemos também que G é um grafo bipartido completo, ou seja,

[3]Discussões mais detalhadas sobre complexidade computacional podem ser encontradas nos textos de Cormen e outros [CLR90], Garey e Johnson [GJ79], Manber [Man89], dentre outros. Cormen e outros, Manber enfatizam os aspectos essenciais relacionados a análise de algoritmos. Garey e Johnson abordam extensivamente a teoria de complexidade computacional.

$|E| = m = n^2$. Esta suposição não restringe o problema visto que arestas com custo suficientemente grande podem ser adicionadas a G quando este não for completo. Note que as arestas adicionadas não deveriam estar presentes em nenhuma solução, a não ser que G não contenha um emparelhamento perfeito com as arestas originais.

Em um grafo bipartido completo existe sempre um emparelhamento perfeito, portanto o PD sempre tem solução. Na verdade existem $n!$ emparelhamentos perfeitos dos quais o melhor corresponde àquele que minimiza o custo. Este emparelhamento será denominado *emparelhamento ótimo.*

Por simplicidade suporemos que os custos das arestas são inteiros não negativos restritos ao intervalo $[0, \ldots, C]$. Esta suposição não é restritiva visto que uma constante suficientemente grande pode ser adicionada a todos os custos sem alterar o emparelhamento ótimo.

O PD pode ser formulado matricialmente com cada célula da matriz indicando o custo para dois elementos se associarem. Um emparelhamento neste caso consiste em escolher n células da matriz, de maneira que haja somente uma célula escolhida em cada linha e coluna. O custo do emparelhamento é o somatório dos custos das células. A solução do PD pode ser expressa por n pares ordenados $(linha, coluna)$.

O PD modela uma variedade de situações que surgem frequentemente em problemas reais. As aplicações vão desde a alocação de recursos até o processamento de sinais. O texto de Ahuja e outros [AMO93] apresentam várias referências onde podem ser obtidas aplicações do problema.

Em aplicações reais podem surgir a necessidade de se resolver sucessivas instâncias do PD que diferem em parte das informações do problema. Uma dessas situações ocorre quando modelamos uma parcela das informações por funções. A cada valor definido pela função está associado uma instância do problema. Neste caso se deseja resolver todas as instâncias da forma mais eficaz possível. Uma maneira para se alcançar tal objetivo é conseguir explorar eficientemente a estrutura do problema a ser resolvido.

Este trabalho dedica-se a estudar o PD quando representamos os custos das arestas em função linear de um parâmetro λ. Esta variante do problema denominada **problema da designação paramétrico** será formalmente descrita no capítulo 5. Neste estudo queremos analisar a estrutura do PD paramétrico e discutir os problemas envolvidos na sua resolução eficiente. Buscamos propôr soluções eficientes computacionalmente, sempre que possível, para o PD paramétrico. No mínimo, espera-se contribuir de alguma forma para enriquecer a literatura sobre o PD paramétrico e sugerir direções na realização de futuros estudos.

O PD além de possuir diversas aplicações na resolução de problemas práticos também está envolvido na resolução de outros problemas mais genéricos. Algoritmos eficientes para a resolução da sua variante paramétrica podem implicar em avanços nos problemas

diretamente modelados pelo PD, como também nos demais que dependem indiretamente da complexidade computacional do problema.

1.4 Organização do texto

Este primeiro capítulo foi destinado a discutir brevemente o problema em estudo nesta dissertação. Nos capítulos subsequentes, o PD paramétrico e todos os demais aspectos envolvidos são abordados mais detalhadamente.

No capítulo 2 discutimos o contexto no qual o PD se insere e o seu relacionamento com outros problemas. No capítulo 3 discutimos os aspectos envolvidos na resolução do PD. Definimos os conceitos aplicados nos algoritmos para o problema e os fundamentos aos quais se baseiam estes algoritmos. No capítulo 4 explicamos as abordagens e os diversos algoritmos aplicados na resolução do problema.

No capítulo 5 abordamos a literatura existente para os problemas paramétricos. Avaliamos as soluções propostas para diversos problemas e finalizamos propondo uma nova abordagem para o PD paramétrico. Discutimos também esta nova abordagem no contexto de um problema mais genérico que o PD paramétrico, o problema do fluxo de custo mínimo paramétrico. No capítulo 6 descrevemos as implementações da abordagem proposta e de outra presente na literatura, ambas para o problema do fluxo de custo mínimo paramétrico, e os resultados obtidos. Finalizamos este texto relatando as contribuições deste estudo.

Em diversos pontos desta dissertação são apresentados teoremas ou lemas cujo entendimento se faz necessário para a abordagem dos assuntos aqui expostos. A quase totalidade desses resultados provêm da literatura, e por isso em geral omitimos uma prova completa. Referências serão apresentadas sempre para a obtenção de informações mais detalhadas sobre estes tópicos.

Capítulo 2

O PD e o seu contexto em Otimização Combinatória

O PD, assim como os demais problemas discutidos neste texto é um problema combinatório. Neste capítulo descrevemos o contexto no qual o PD se insere, assim como a sua importância e a do seu relacionamento com outros problemas combinatórios.

2.1 Otimização Combinatória

Otimização Combinatória tem merecido nas últimas décadas o interesse de diversos estudiosos em diferentes ramos. O motivo de tal interesse vem do fato de que os problemas em Otimização Combinatória têm se mostrado bastante abrangentes, sendo capazes de modelar uma variedade considerável de situações reais.

A natureza desses problemas envolve a busca por uma solução que sob algum critério, especificado por uma função de custo associada, se destaca em relação às demais. Os problemas são combinatórios no sentido de que a *solução procurada*, que usualmente chamamos de **solução ótima**, faz parte de um conjunto de soluções possíveis denominadas *soluções factíveis*. Este conjunto engloba todas as soluções que satisfazem todas as restrições do problema. Em princípio estes problemas podem ser resolvidos enumerando todas as soluções factíveis e determinando-se a melhor. Entretanto, computacionalmente tal alternativa se torna em quase a totalidade dos casos impraticável. O número de soluções factíveis geralmente é exponencial.

Os ramos de aplicação da Otimização Combinatória englobam as mais diversas áreas do conhecimento. Os textos de Gondran e Minoux [GM84], Nemhauser e Wolsey [NW89], Papadimitriou e Steiglitz [PS98] abordam largamente o assunto.

Os problemas em Otimização Combinatória podem ser formulados frequentemente como instâncias de Programação Inteira ou Programação Linear. **Programação Inteira**

e **Programação Linear** são dois modelos matemáticos que se caracterizam por serem descritos por um conjunto de restrições lineares e uma função de custo associada. O objetivo em cada um desses modelos é determinar uma solução que satisfaça todas as restrições e que minimize (ou maximize) a função de custo associada. A diferença conceitual básica nos dois modelos reside no fato de que em Programação Inteira existe uma restrição adicional exigindo a integralidade da solução ótima.

Computacionalmente, os dois modelos diferem por terem complexidades bem distintas. Para a Programação Linear algoritmos eficientes existem, maiores informações no texto de Nemhauser e Wolsey [NW89]. Por outro lado, a Programação Inteira consistirá num modelo computacionalmente muito difícil se a condição $P \neq NP$ se verificar. Karp [Kar72], Borosh e Treybig [BT76] exibem resultados que comprovam que a Programação Inteira (além de algumas variantes) é um problema *NP*-Completo. A complexidade se estende até mesmo ao caso particular de Programação Inteira $0/1$[1].

Casos particulares importantes de Programação Inteira são os problemas de Fluxos em Redes. Os problemas de **Fluxos em Redes** se caracterizam por envolverem a distribuição de fluxo entre os vértices de um grafo, respeitando as possíveis restrições existentes sobre os vértices e as arestas. A forma como o fluxo deve ser distribuído depende fundamentalmente do contexto do problema em estudo.

Os problemas de Fluxos em Redes são largamente aplicados na modelagem de: problemas de produção, distribuição e transporte; problemas de políticas públicas e sociais; problemas em ciências médicas e físicas, dentre outros. Os problemas citados são apenas uma parcela do espectro de aplicações existentes. Ahuja e outros [AMO93] abordam diversos problemas de Fluxos em Redes enfatizando as técnicas frequentemente empregadas, o relacionamento entre os diversos problemas além de uma extensa lista de aplicações.

Um dos problemas mais simples e importante de Fluxos em Redes corresponde ao PD. Apesar de termos formulado o problema anteriormente como um relacionamento entre elementos, o PD pode ser facilmente descrito como sendo um problema de Fluxos em Redes. O PD desperta um interesse grande, tanto prático quanto teórico, nos pesquisadores. Prático porque o problema surge em situações reais bem diversificadas. Teórico porque apesar de ser um dos modelos mais simples, alguns problemas importantes de Fluxos em Redes podem ser convertidos no PD. Instâncias do problema também aparecem naturalmente na resolução de diversos problemas consideravelmente mais complexos, alguns dos quais sendo problemas *NP*-Difíceis.

[1] Garey e Johnson [GJ79] discutem detalhadamente a teoria de *NP*-Completude, além de enfatizar os conceitos e técnicas aplicadas à resolução de problemas intratáveis como os problemas *NP*-Completos

A formulação do PD como um problema de Programação Inteira é descrita a seguir:

$$\min \sum_{(x,y)\in E} c_{xy} f_{xy}$$

Sujeito a:

$$\sum_{y\in Y} f_{xy} = 1 \qquad \forall x \in X \qquad \text{(I)}$$

$$\sum_{x\in X} f_{xy} = 1 \qquad \forall y \in Y \qquad \text{(II)}$$

$$f_{xy} \in \{0,1\} \qquad \forall (x,y) \in E \quad \text{(RI)}$$

A variável f_{xy} indica que a aresta (x, y) pertence ao emparelhamento se $f_{xy} = 1$, caso contrário $f_{xy} = 0$. A restrição I assegura que existe exatamente uma aresta incidente ao vértice $x \in X$ em qualquer solução ótima. A restrição II expressa a mesma condição para os vértices da partição Y. Nesta descrição é imposto pela restrição RI que a variável $f_{xy} \in \{0, 1\}$. Esta restrição é denominada *restrição de integralidade.*

Os algoritmos que resolvem o PD exploram as características especiais do problema para a obtenção de soluções eficientes, ao invés de tratar o PD simplesmente como uma instância de Programação Inteira.

2.2 O problema do fluxo de custo mínimo

Seja $G = (V, E)$ um grafo orientado. A cada aresta $e \in E$ está associado um custo c_e não negativo para cada unidade de fluxo atravessando a aresta. Limites máximo u_e e mínimo l_e indicam a quantidade de fluxo permitido para a aresta. Associamos a cada vértice v um valor b_v para representar o seu excesso ou demanda por fluxo. O **problema do fluxo de custo mínimo**, o qual será denotado por PFCM, consiste em rotear com o menor custo possível o fluxo através de G, respeitando as restrições sobre as arestas com o objetivo de satisfazer o excesso/demanda de fluxo nos vértices. Sem perda de generalidade suporemos que os custos, as capacidades das arestas, o excesso/demanda de fluxo dos vértices são inteiros.

O PFCM é o modelo mais importante de Fluxos em Redes. Os demais problemas de Fluxos em Redes, como o *problema da designação*, o *problema dos caminhos mais curtos* e o *problema do fluxo máximo*, podem ser interpretados como casos particulares do PFCM. O PFCM surge em uma variedade de situações como gerenciamento de recursos humanos, planejamento de sistemas de distribuição, escalonamento, diagnósticos médicos, dentre outras[2].

[2]Estudos sobre a estrutura do PFCM e suas aplicações são abordados por Ahuja, Magnanti e Orlin [AMO93], Bertsekas [Ber91] e Tarjan [Tar83]

Formalmente o PFCM pode ser descrito como o seguinte problema de Programação Inteira:

$$\min \sum_{(v,w)\in E} c_{vw} f_{vw}$$

Sujeito a:

$$\sum_{w\in V} f_{vw} = b_v + \sum_{w\in V} f_{wv} \qquad \forall v \in V \qquad \text{(RCF)}$$

$$l_{vw} \leq f_{vw} \leq u_{vw} \qquad \forall (v,w) \in E$$

$$f_{vw} \in Z_+^n \qquad \forall (v,w) \in E \qquad \text{(RI)}$$

A restrição RI é a *restrição de integralidade* para o PFCM. A restrição RCF é denominada *restrição de conservação de fluxo*. Esta restrição indica que o sistema é conservativo, ou seja, a quantidade de fluxo total no sistema permanece a mesma. Um **pseudofluxo** é uma relaxação da condição de conservação de fluxo onde se admite vértices desbalanceados. O *desbalanceamento* de um vértice é definido como:

$$D(v) = b_v + \sum_{v\in V} f_{wv} - \sum_{v\in V} f_{vw}$$

Os vértices são classificados de acordo com o seu desbalanceamento. Os vértices com $D(v) > 0$ são definidos vértices com *excesso* de fluxo. Os vértices com $D(v) < 0$ são definidos vértices com *demanda* por fluxo. Referimos aos demais vértices com $D(v) = 0$ como vértices *balanceados*.

O melhor limite de tempo existente atualmente para a resolução do PFCM é $O(\min\{(m \log n)(m + n \log n),\ nm \log(\frac{n^2}{m}) \log(nC),\ nm(\log\log U)\log(nC))\}$. O limite de tempo polinomial forte é definido pelo algoritmo de Orlin [Orl88]. O limite de tempo $O(nm \log(\frac{n^2}{m}) \log(nC))$ é devido ao algoritmo de Goldberg e Tarjan [GT90]. O limite de tempo $O(nm(\log\log U)\log(nC))$ é devido ao algoritmo de Ahuja e outros [AGOT92].

Um dos principais resultados de Fluxos em Redes demonstra a estrutura particular destes problemas. A matriz correspondente às restrições presentes na formulação matemática do PFCM como uma instância de *PL*, é totalmente unimodular. Como consequência, qualquer instância do PFCM com excesso/demanda de fluxo nos vértices inteiros e as capacidades das arestas inteiras possui uma solução ótima com fluxos inteiros nas arestas. O texto de Ahuja e outros [AMO93] apresenta diversas referências que podem ser consultadas para se obter uma demonstração formal e uma análise em profundidade desta propriedade, e sobre tópicos relacionados a unimodularidade.

2.3 Relação entre o PD e o problema do fluxo de custo mínimo

O PD foi descrito anteriormente como um problema de emparelhamento em grafos. Nesta seção, discutiremos como o problema pode ser interpretado como um problema de fluxo. Além disso, discutiremos importantes relações entre o PD e outros problemas de Fluxos em Redes.

O PD pode ser reduzido ao PFCM realizando a seguinte transformação.

Redução do PD ao PFCM Dada uma instância $G = (V, E)$ do PD com partições $V = X \bigcup Y$ e arestas $e \in E$ de custo c_e, defina uma instância $G' = (V', E')$ do PFCM com o conjunto de vértices $V' = V$. Para cada aresta $e \in E$ defina uma aresta e' em G' com custo $c_{e'} = c_e$ e capacidade $u_{e'} = 1$. Insira dois vértices especiais s e t em V' e adicione as arestas $e' = (s, x)\ \forall x \in X$ e $e' = (y, t)\ \forall y \in Y$ com custo $c_{e'} = 0$ e capacidade $u_{e'} = 1$. Defina o fluxo b a ser enviado de s para t como sendo $b = |X|$.

Observe que sob a perspectiva de um problema de fluxo as arestas em um emparelhamento devem ser saturadas ou devem ter fluxo nulo. As arestas *saturadas* são as arestas com fluxo igual à sua capacidade. As arestas emparelhadas em um grafo são as arestas saturadas. Desta maneira, o emparelhamento ótimo de G pode ser obtido a partir da solução ótima do PFCM em G' observando quais arestas estão saturadas.

A redução do PD ao PFCM mostrada acima é simples. O PD participa de outra redução que é igualmente importante e não tão simples que será descrita a seguir:

O problema de transporte, abreviado por PT, é um caso particular do PFCM onde o grafo é bipartido. Formalmente podemos descrever o PT como a seguir:

Seja $G = (V, E)$ um grafo bipartido orientado com partições $V = X \bigcup Y$. Os vértices de X são as *fontes* e os vértices de Y os *sorvedouros*. Cada fonte x possui um excesso inteiro $b_x > 0$ que deve ser distribuído para satisfazer a demanda nos sorvedouros. A demanda no sorvedouro y é um inteiro $b_y < 0$. Para cada unidade de fluxo atravessando a aresta $e \in E$ está associado um custo c_e e uma capacidade u_e. O sistema está em equilíbrio no sentido de que $\sum_{x \in X} b_x = \sum_{y \in Y} b_y$. O *problema de transporte* consiste em rotear, com o menor custo possível, o fluxo através de G de maneira a balancear os vértices.

O PT promove um importante vínculo entre o PFCM e o PD. Apesar do PT ser um caso particular do PFCM, pode-se transformar o PFCM no PT. Por sua vez, o próprio PT pode ser convertido no PD. Entretanto, a redução em questão é pseudopolinomial. Desta maneira, apesar de ser um dos modelos mais simples o PD possui uma estrutura peculiar que permite modelar problemas mais genéricos, como é o caso do PFCM. As reduções do

PFCM ao PT e do PT ao PD são descritas a seguir[3]:

Redução do PFCM ao PT [4] Dada uma instância $G = (V, E)$ do PFCM construa uma instância $G' = (V', E')$ do PT realizando as seguintes transformações. Para cada aresta $e = (v, w)$ em G insira três vértices e', v' e w' em G'. Conecte estes vértices adicionando as arestas orientadas (e', v') e (e', w'). Defina o custo da aresta (e', v') como sendo $c_{e'v'} = 0$. Defina o custo da aresta (e', w') como sendo $c_{e'w'} = c_{vw}$, como indicado na figura 2.1. O fluxo através da aresta (e', v') é indicado por z_{vw} e o fluxo através da aresta (e', w') por x_{vw}.

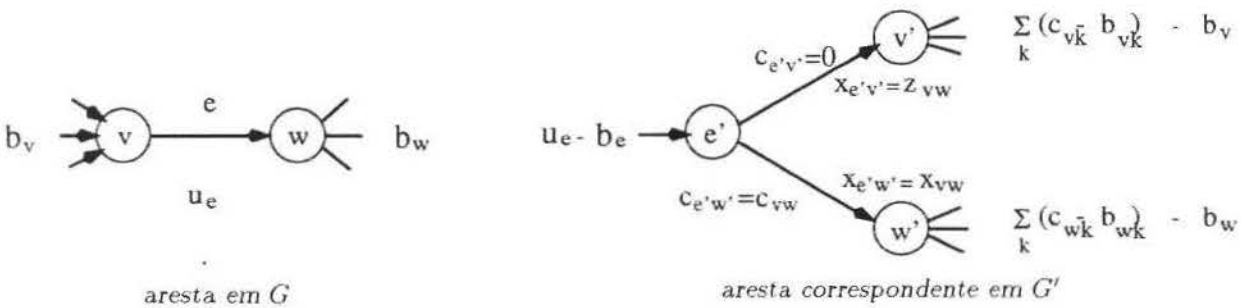


Figura 2.1: *Transformação do* PFCM *no* PT

Para que o PFCM tenha uma solução factível devemos ter $u_e = z_{vw} + x_{vw}$. Neste caso, a restrição de capacidade $x_{vw} \leq u_e$ em G é equivalente a restrição $0 \leq z_{vw}$ em G'. Pode ser demonstrado que esta transformação elimina a restrição de limite superior para as arestas além de criar um grafo bipartido. A restrição de conservação de fluxo em G'

$$-\sum_{w \in V} z_{vw} - \sum_{w \in V} x_{wv} = b_v - \sum_{w \in V} u_{vw}$$

é equivalente a restrição de G:

$$\sum_{w \in V} f_{vw} - \sum_{w \in V} f_{wv} = b_v$$

A partir destes fatos pode ser provado que o grafo resultante é equivalente ao original. As arestas de G correspondem às fontes em G'. Uma fonte e' possui excesso de fluxo igual à quantidade de fluxo que é possível enviar através da aresta correspondente em G, $b_{e'} = u_e - l_e$. Por sua vez, os vértices de G corresponderam aos sorvedouros em G'. A demanda de um sorvedouro v' é a soma das quantidades de fluxo que se pode

[3] Uma análise mais detalhada sobre a relação entre os problemas de Fluxos em Redes pode ser obtida no texto de Balakrishnan [Bal95]. Balakrishnan apresenta uma extensa discussão sobre reduções entre os problemas de Fluxos em Redes

[4] A redução apresentada é baseada no texto de Bertsekas [Ber91]

enviar através das arestas partindo do vértice menos o seu excesso (veja figura 2.1). O fluxo de cada aresta e de G na solução ótima do PFCM é a soma dos fluxos das arestas (e', v') e (e', w') em G' na solução ótima do PT, $f_e = x_{vw} + z_{vw}$. O fluxo na aresta e é $f_e = x_{vw} + u_{vw} - l_{vw} - x_{vw}$.

Redução do PT ao PD [5] Seja $G = (V, E)$ uma instância do PT com partições $V = X \bigcup Y$. Defina uma instância G' do PD realizando as seguintes transformações. Para cada fonte $x \in X$ com excesso de fluxo b_x insira em G' os vértices x_α, $\alpha = 1, \ldots, b_x$. Para cada sorvedouro $y \in Y$ com demanda b_y insira em G' os vértices y_β, $\beta = 1, \ldots, b_y$. Para cada aresta orientada (x, y) em G adicione em G' as arestas não orientadas (x_α, y_β), $\alpha = 1, \ldots, b_x$ e $\beta = 1, \ldots, b_y$, todas com custos $c_{x_\alpha y_\beta} = c_{xy}$ (veja figura 2.2). Nesta transformação a bipartição do grafo é mantida.

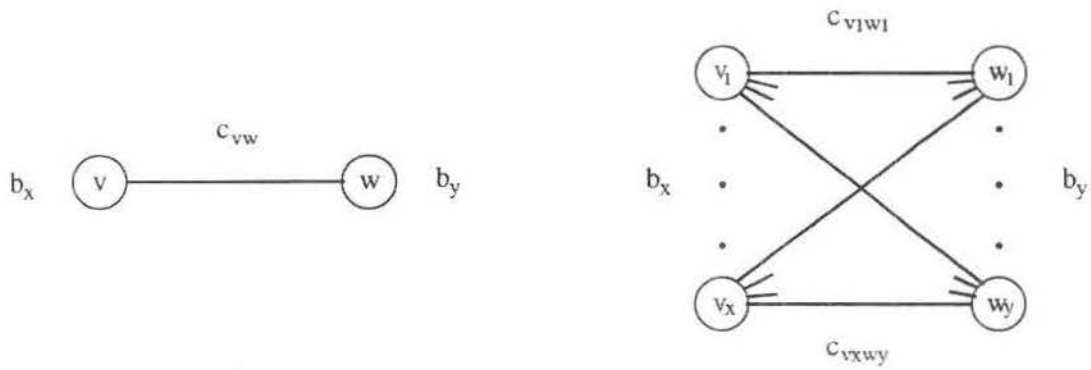


Figura 2.2: *Transformação do* PT *no* PD

A solução ótima do PT pode ser obtida através do emparelhamento ótimo de G'. Se uma aresta (x_α, y_β) está presente no emparelhamento ótimo temos uma unidade de fluxo na aresta (x, y) em G. O fluxo total na aresta (x, y) é determinado pela quantidade de arestas (x_α, y_β) no emparelhamento ótimo de G'.

Por ser um problema mais simples, o PD pode ser utilizado para melhorar a nossa compreensão sobre a estrutura de problemas mais genéricos como o PFCM. Novas técnicas desenvolvidas para os problemas de Fluxos em Redes, podem ser aplicadas primeiramente ao PD para uma avaliação de sua eficiência. Posteriormente, estas técnicas podem ser generalizadas para o desenvolvimento de algoritmos para problemas mais genéricos. Um exemplo clássico desta abordagem é o *algoritmo primal-dual*, também conhecido como *algoritmo húngaro*. O algoritmo primal-dual desenvolvido por Ford e Fulkerson [FF57]

[5] A redução apresentada é baseada no texto de West [Wes96]

primeiramente para o PT, e posteriormente estendido para o PFCM em [FF62], foi generalizado a partir do algoritmo primal-dual proposto por Kuhn [Kuh55, Kuh56] para o PD.

2.4 O problema do caixeiro viajante

Instâncias do PD surgem naturalmente na resolução de diversos problemas combinatórios difíceis, como por exemplo: o *problema do caixeiro viajante* [LLKS85], o *problema da designação quadrático* [PRW93], o *problema de roteamento de veículos* [LN87], o *problema de rotação de tripulação* [BGAB83], dentre outros. Todos os problemas citados são conhecidamente problemas *NP*-Difíceis.

O **problema do caixeiro viajante** -denotado por PCV- é um dos problemas mais estudados em Otimização Combinatória. Seja $G = (V, E)$ um grafo orientado com arestas $e \in E$ de custo c_e. O PCV consiste em determinar um ciclo hamiltoniano de custo mínimo em G. Um *ciclo hamiltoniano* em G é um ciclo onde todo vértice $v \in V$ está presente uma única vez.

Representação Matricial						
	1	2	3	4	5	6
1	0	10	∞	∞	∞	∞
2	∞	0	9	∞	∞	6
3	14	∞	0	12	∞	∞
4	∞	∞	∞	0	4	∞
5	∞	15	-3	∞	0	-4
6	-5	∞	∞	8	∞	0

Tabela 2.1: *Representação matricial de uma instância do* PD *e* PCV

Da mesma maneira como o PD, o PCV também pode ter formulação matricial (veja a tabela 2.1). Neste caso, cada célula da matriz $M[ij]$ indica o custo da aresta dirigida (i, j). As mesmas restrições quanto à solução do PD se aplicam ao PCV com uma restrição adicional. Os pares ordenados que expressam a solução do problema, devem corresponder à uma permutação cíclica de todos os vértices do problema. A solução do PD não necessariamente precisa possuir esta característica. Os pares ordenados na solução do PD vão corresponder à uma coleção de ciclos orientados disjuntos cujo custo total de todos os ciclos da coleção é mínimo.

Observe que os pares ordenados que representam ciclo hamiltoniano são soluções factíveis para o PD, pois estes pares equivalem a emparelhamentos perfeitos em um grafo

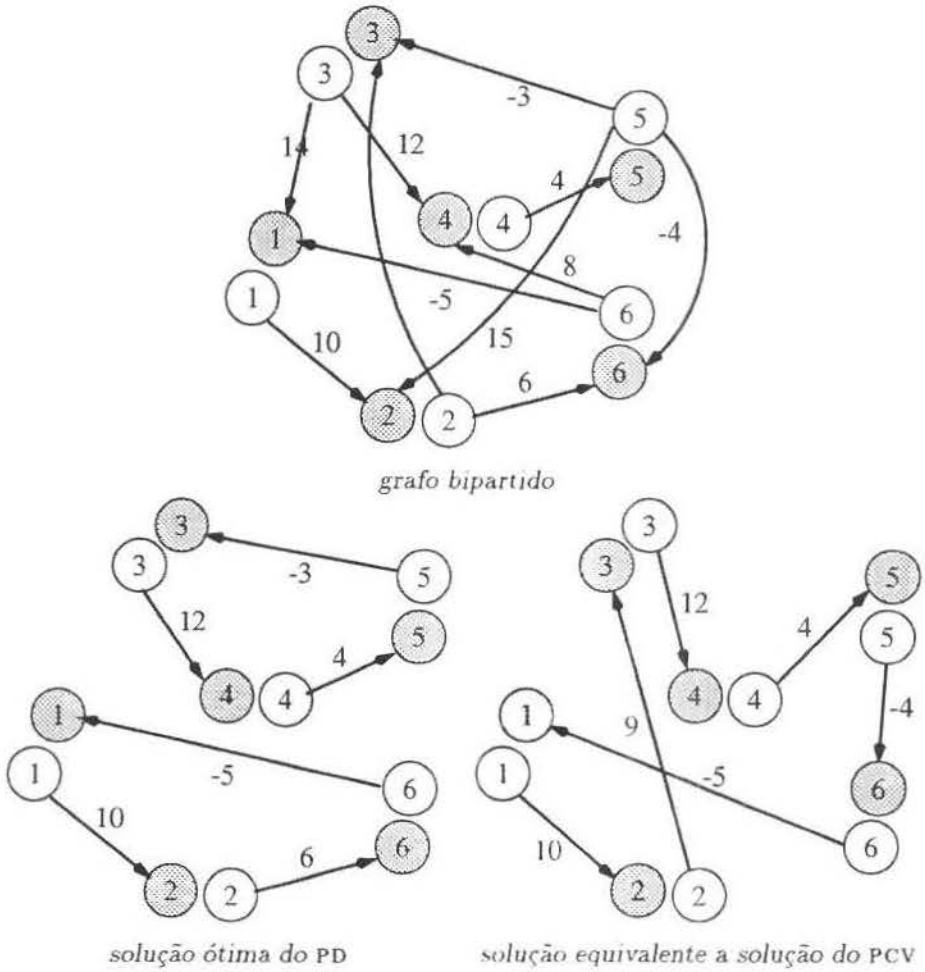


Figura 2.3: *Instâncias do* PD

bipartido. Na verdade, das $n!$ possíveis soluções factíveis para o PD $(n-1)!$ são permutações cíclicas de todos os vértices e portanto equivalem à ciclos hamiltonianos em um grafo orientado. O PD pode ser visualizado como uma relaxação do PCV no sentido de que ciclos (não hamiltonianos) são soluções factíveis (veja figura 2.3 e 2.4).

Pelo exposto, o custo da solução ótima do PD é um limite inferior para o custo da solução ótima do PCV. Lawler e outros [LLKS85] relatam que frequentemente o limite fornecido pelo PD está relativamente próximo da solução ótima do PCV. Pode-se obter uma formulação para o PCV, inserindo-se restrições adicionais à definição do PD para excluir os ciclos não hamiltonianos do conjunto de soluções factíveis. A restrição descrita a seguir denominada *restrição de eliminação de ciclos* tem esta finalidade.

$$\sum_{\substack{(v,w)\in E \\ v\in U,\ w\in U}} f_{vw} \leq |U| - 1 \qquad 2 \leq |U| \leq |V| - 2 \quad \text{(REC)}$$

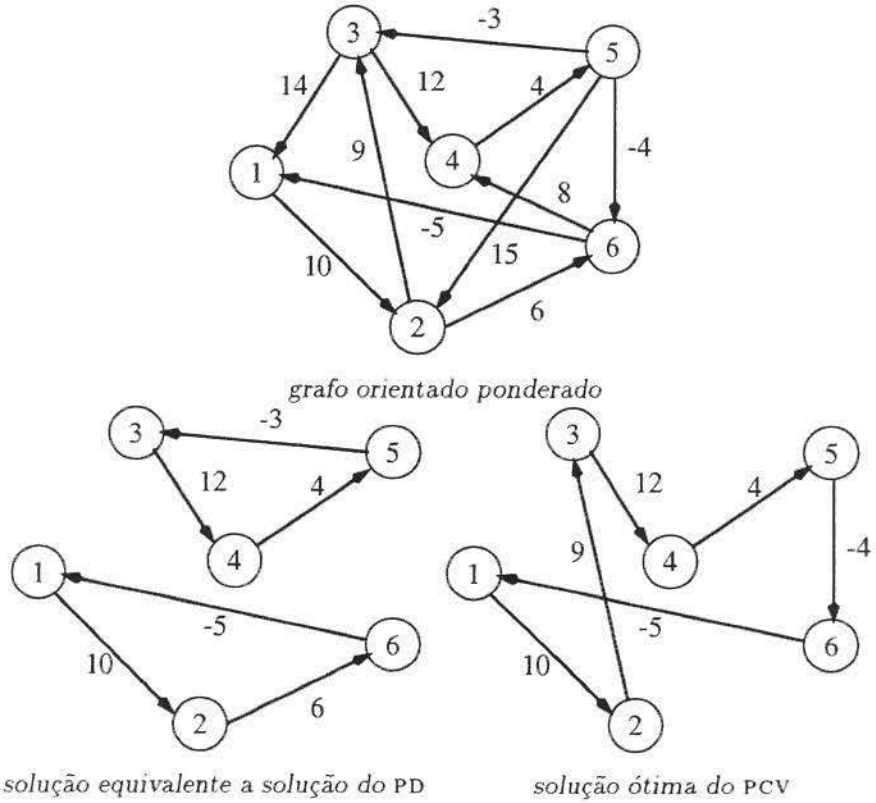


Figura 2.4: *Instâncias do* PCV

A restrição REC deve ser definida para cada subconjunto $U \subset V$ de vértices. A restrição exige que cada subconjunto de vértices U contenha no máximo $|U| - 1$ arestas, o que garante que nenhum subconjunto de vértices U contém um ciclo. Esta restrição somente é satisfeita por ciclos hamiltonianos. Para qualquer outro tipo de solução haverá no mínimo uma restrição REC sendo violada.

A complexidade do PCV está presente exatamente na restrição REC. Ao contrário do PD, o PCV é um problema *NP*-Difícil. A importância do PD neste contexto vem da possibilidade de utilizarmos o problema como uma importante relaxação do PCV. Historicamente o PD foi a primeira relaxação utilizada para a resolução do PCV, por causa da simplicidade da relaxação e da facilidade envolvida na computação do PD.

Algumas metodologias aplicadas a problemas complexos, como o método *branch-and-bound*, envolvem a resolução de uma série de problemas mais simples. O desempenho destas metodologias deveria ser afetada indiretamente por avanços para o PD quando aplicadas aos problemas descritos no início da seção. Mais informações sobre a utilização do PD nestas situações podem ser encontradas em Miller e Pekny [MP89].

A relação entre o PD e o PCV se estendem também às variantes do problema. Um caso particular onde o PCV pode ser resolvido eficientemente é o PCV com matriz de custos

triangular. Nesta variante, a representação matricial do problema é uma matriz triangular pois o custo das arestas de retorno, arestas do tipo (v_i, v_j) sendo $i \geq j$, é nulo. O custo das demais arestas do grafo é arbitrário.

Lawler [Law71] resolveu esta variante eficientemente através da resolução de uma instância do PD. O algoritmo consiste em obter um emparelhamento ótimo na matriz resultante da exclusão da primeira coluna e da última linha. Se o emparelhamento obtido mais a aresta (v_n, v_1) é uma permutação cíclica dos vértices, este emparelhamento equivale à um ciclo hamiltoniano de custo mínimo. Caso contrário, o emparelhamento equivale à uma coleção de ciclos disjuntos de custo mínimo. Se todas as arestas de retorno forem eliminadas desta coleção o resultado será uma coleção de caminhos da forma $(v_{i^1}, v_{j^1}), \ldots, (v_{i^k}, v_{j^k}), \ldots$ sendo $i^1 < i^2, \ldots, i^{k-1} < i^k$ e $j^1 < j^2, \ldots, j^{k-1} < j^k$. Insira as seguintes arestas de retorno $(v_{j^1}, v_{i^2}), (v_{j^2}, v_{i^3}), \ldots, (v_{j^k}, v_{i^1})$ conectando os vários caminhos para formar um ciclo hamiltoniano. Como todas as arestas excluídas e inseridas são arestas de retorno, o ciclo hamiltoniano obtido possui o mesmo custo do emparelhamento ótimo.

Capítulo 3

Conceitos para os problemas de emparelhamento

Neste capítulo, serão abordados os conceitos e resultados mais importantes de Fluxos em Redes relacionados aos problemas de emparelhamentos. Na resolução do PD naturalmente surgem instâncias de outros problemas de Fluxos em Redes. Neste capítulo, estes problemas serão discutidos para obtermos uma compreensão melhor das características do PD e dos fundamentos nos quais se baseiam os algoritmos descritos no capítulo 4. Uma análise mais detalhada pode ser obtida no texto de Ahuja e outros [AMO93].

3.1 Custos reduzidos

Em determinadas situações é importante representar o custo de uma aresta em relação aos vértices incidentes a mesma. Para isto necessitamos associar valores aos vértices.

Seja $G = (V, E)$ uma instância do PFCM. Considere associado a cada vértice $v \in V$ um número $\pi(v)$ o qual referiremos como o seu **potencial**[1]. Definimos o **custo reduzido** da aresta (v, w) em relação aos potenciais π como sendo $c^{\pi}_{vw} = c_{vw} - \pi(v) + \pi(w)$.

Os potenciais dos vértices não alteram o menor caminho entre algum par de vértices v e w. Para algum caminho orientado P do vértice v ao vértice w temos que

$$\begin{aligned}\sum_{(i,j)\in P} c^{\pi}_{ij} &= \sum_{(i,j)\in P} [c_{ij} - \pi(i) + \pi(j)] \\ &= \sum_{(i,j)\in P} c_{ij} - \sum_{(i,j)\in P} [\pi(i) - \pi(j)] \\ &= \sum_{(i,j)\in P} c_{ij} - \pi(v) + \pi(w)\end{aligned}$$

[1] Os *potenciais* dos vértices são as variáveis duais da Programação Linear

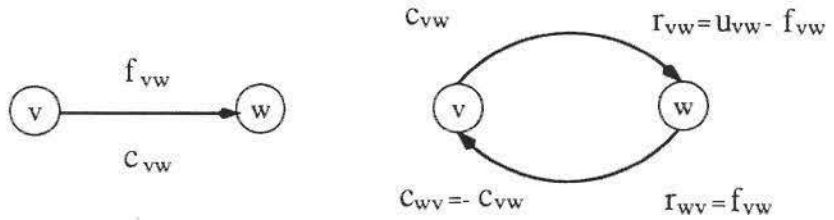


Figura 3.1: *Grafo residual*

Deste modo os potenciais aumentam o tamanho de cada caminho por um fator constante $[\pi(w) - \pi(v)]$. Esta propriedade também implica que se $\mathcal{C}$ é um ciclo de custo negativo com respeito aos custos das arestas, o mesmo ocorre com respeito aos custos reduzidos. Para algum ciclo orientado $\mathcal{C}$ temos que

$$\sum_{(i,j)\in\mathcal{C}} c_{ij}^{\pi} = \sum_{(i,j)\in\mathcal{C}} [c_{ij} - \pi(i) + \pi(j)]$$
$$\sum_{(i,j)\in\mathcal{C}} c_{ij}^{\pi} = \sum_{(i,j)\in\mathcal{C}} c_{ij}$$

Os custos reduzidos não descaracterizam as relações básicas entre os vértices, como por exemplo, caminhos e ciclos. Pelo contrário, os custos reduzidos possibilitam uma compreensão melhor do comportamento das soluções dos problemas. Algumas condições importantes para caracterizar a otimalidade de uma solução utilizam o conceito de custo reduzido.

3.2 Grafo residual

Para simplificar a exposição do conceito de grafo residual realizaremos uma transformação no grafo. A transformação não afetará a estrutura do problema ma apenas a sua representação. O conjunto de vértices será mantido. Cada aresta (v, w) será substituída por duas arestas (v, w) e (w, v) com a mesma capacidade tal que $c_{vw} = -c_{wv}$, como ilustrado na figura 3.1. A função da aresta reversa (w, v) é possibilitar o retorno para v do fluxo enviado para w através de (v, w). Intuitivamente o fluxo em (w, v) anula o fluxo em (v, w), ou seja, $f_{wv} = u_{vw} - f_{vw}$. A presença de arestas múltiplas pode ser manipulada algoritmicamente sem alterações na complexidade do problema.

Em problemas de Fluxos em Redes é conveniente representar cada solução em termos do fluxo restante em cada aresta. O objetivo é poder expressar quanto de fluxo adicional pode ser enviado através de cada aresta. Com uma representação mais apropriada será possível estabelecer como a solução corrente pode ser alterada.

Para definir o grafo residual formalmente será necessário introduzir o conceito de capacidade residual. A **capacidade residual** de uma aresta é a quantidade de fluxo adicional que pode ser enviado através da mesma. A capacidade residual da aresta (v, w) é $r_{vw} = u_{vw} - f_{vw}$ e de sua reversa (w, v) é $r_{wv} = f_{vw}$.

Pode-se definir o **grafo residual** $G(\mathcal{F}) = (E(\mathcal{F}), V(\mathcal{F}))$ associado à um fluxo $\mathcal{F}$ como sendo o grafo contendo apenas arestas com capacidade residual positiva. A definição de grafo residual é válida tanto em relação aos custos das arestas quanto em relação aos custos reduzidos.

No grafo residual de um emparelhamento somente existe uma aresta entre cada par de vértices. O grafo residual permite assim distinguir claramente as arestas emparelhadas das não-emparelhadas. As arestas emparelhadas correspondem as arestas reversas do grafo residual.

3.3 Emparelhamento de cardinalidade máxima

Os problemas de emparelhamentos (ponderado ou não) são mais fáceis de investigar no caso particular onde o grafo é bipartido. O estudo do caso bipartido pode resultar eventualmente em generalizações apropriadas para o caso não bipartido. Nesta seção, discutiremos aspectos do problema de emparelhamento bipartido não ponderado que estão relacionados ao PD.

O problema não ponderado conhecido como o **problema de emparelhamento de cardinalidade máxima**, denotado por PECM, tem o objetivo de obter um emparelhamento com a maior cardinalidade possível em um grafo[2].

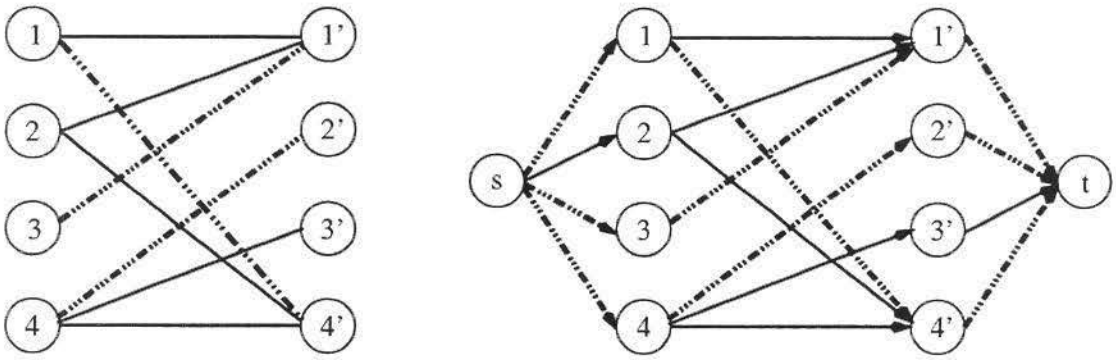
Uma resolução natural do PECM é reduzi-lo ao *problema do fluxo máximo*, denotado por PFM. Uma simples metodologia para resolver o PFM resulta em importantes conceitos para a teoria de emparelhamentos. A abordagem destes conceitos é o propósito desta seção. Antes de discutirmos a relação entre estes problemas descreveremos o problema do fluxo máximo.

Seja $G = (V, E)$ um grafo orientado com dois vértices específicos uma fonte s e um sorvedouro t. Para cada aresta $e \in E$ existe um limite máximo u_e não negativo de fluxo que pode atravessar a aresta. O PFM consiste em enviar o máximo possível de fluxo da fonte para o sorvedouro respeitando a capacidade das arestas.

O PFM pode ser formulado como uma instância do PFCM estabelecendo $b_v = 0\ \forall v \in V$, $c_{vw} = 0\ \forall (v, w) \in E$, inserindo um aresta (t, s) com custo $c_{ts} = -1$ e capacidade $u_{ts} = \infty$. A solução ótima do PFM é o fluxo enviado de t para s na solução ótima do PFCM.

Entretanto, o PFM pode ser resolvido de maneira consideravelmente mais eficiente que o PFCM. O melhor limite de tempo para a resolução do PFM é definido pelo algoritmo de

[2]Informações adicionais podem ser obtidas no texto de Cormen e outros [CLR90]



emparelhamento de cardinalidade máxima *fluxo máximo correspondente*

Figura 3.2: *Redução do* PECM *ao* PFM

Goldberg e Rao [GR98] $O(\min\{\sqrt[3]{n^2}, \sqrt{m}\} m \log(\frac{n^2}{m}) \log U\})$[3].

A redução do PECM ao PFM é descrita a seguir:

Redução do PECM ao PFM [4] Seja $G = (V, E)$ uma instância do PECM com partições X e Y. Construa uma instância do PFM realizando as seguintes transformações. Primeiramente obtenha uma versão orientada G' de G com arestas no sentido de X para Y. Insira uma fonte s e um sorvedouro t em G'. Adicione arestas da fonte aos vértices de X e arestas dos vértices de Y ao sorvedouro em G'. Defina a capacidade de todas as arestas como sendo unitária (veja figura 3.2).

Observe que todos os caminhos entre s e t têm capacidade máxima unitária. A maior quantidade de fluxo que se pode enviar de s para t através de um único caminho é uma unidade. Se o valor do fluxo máximo em G' é ψ, devem existir exatamente ψ caminhos disjuntos entre s e t. Cada caminho entre s e t contém uma aresta entre as partições X e Y. Porque os caminhos são disjuntos, as arestas saturadas na solução ótima do PFM formam um emparelhamento de cardinalidade máxima ψ.

Um dos métodos mais simples para a resolução do PFM baseia-se na operação de *aumento*. A operação consiste em a partir de um fluxo inicial no grafo (que pode ser nulo), aumentar continuamente o fluxo enviado da fonte para o sorvedouro. Devido às restrições nas capacidades das arestas este processo é finito. O número de operações, entretanto, não é limitado polinomialmente.

Um dos conceitos importantes neste processo é o conceito de caminho aumentante. Um caminho entre s e t é *aumentante* se cada aresta no caminho possui capacidade

[3] Ahuja, Magnanti e Orlin [AMO93], Lawler [Law76], Papadimitriou e Steiglitz [PS98], Tarjan [Tar83] apresentam aplicações e os aspectos relacionados ao PFM mais detalhadamente

[4] A redução apresentada é baseada no texto de Ahuja e outros [AMO93]

residual positiva. Os caminhos aumentantes possibilitam caracterizar de forma simples o fluxo máximo em um grafo. Um fluxo no grafo somente pode ser máximo se não existem caminhos aumentantes entre s e t.

A simplicidade destes conceitos motivou a especialização dos mesmos para os problemas de emparelhamentos. O conceito de caminho aumentante pode ser aplicado aos vários problemas de emparelhamentos. No caso do problema de emparelhamento bipartido o conceito torna-se extremamente simples[5].

3.3.1 Caminhos aumentantes

Um caminho P é *alternante* se este é uma sequência alternando arestas emparelhadas e não emparelhadas. Pode-se definir caminhos alternantes pares e ímpares baseando-se no número de arestas do caminho. Um *ciclo alternante* é um caminho alternante que inicia e finaliza no mesmo vértice.

Por ser bipartido, os caminhos no grafo residual $G(\mathcal{M})$ devem alternar entre os vértices das partições X e Y. Recorde que as arestas emparelhadas são as arestas reversas no grafo residual. Os caminhos em $G(\mathcal{M})$ são formados por uma sequência alternando arestas emparelhadas e não emparelhadas. Isto implica que qualquer caminho em $G(\mathcal{M})$ é um caminho alternante. Pelo mesmo motivo, se existe um ciclo em $G(\mathcal{M})$ este deve ser um caminho alternante que inicia e finaliza no mesmo vértice, ou seja, um ciclo alternante.

Uma *árvore alternante* $\mathcal{T}$ com raiz v relativa a um emparelhamento $\mathcal{M}$ é um subgrafo parcial de $G(\mathcal{M})$ acíclico, tal que:

i) v é um vértice não-emparelhado;

ii) $\forall w \in \mathcal{T}$, o único caminho na árvore entre v e w é o caminho alternante com o menor número de arestas entre eles.

Pelas características dos caminhos de $G(\mathcal{M})$, uma árvore alternante é a árvore de caminhos mais curtos. Neste caso, a árvore alternante pode ser obtida em tempo polinomial.

Definimos um **caminho aumentante** relativo à um emparelhamento $\mathcal{M}$ como sendo um caminho alternante ímpar onde o primeiro e o último vértice não estão emparelhados. A importância desta nova definição é que os caminhos alternantes são mais intuitivos e adequados à teoria de emparelhamento.

A importância do conceito de caminho aumentante na teoria de emparelhamento vem do teorema descrito a seguir provado por Berge [Ber57], Norman e Rabin [NR59].

Teorema 3.3.1 *Um emparelhamento $\mathcal{M}$ é de tamanho máximo se e somente se ele não contém nenhum caminho aumentante.*

[5] Balakrishnan [Bal95] discute diversas implicações e equivalências dos teoremas de Fluxos em Redes e em Teoria de Grafos

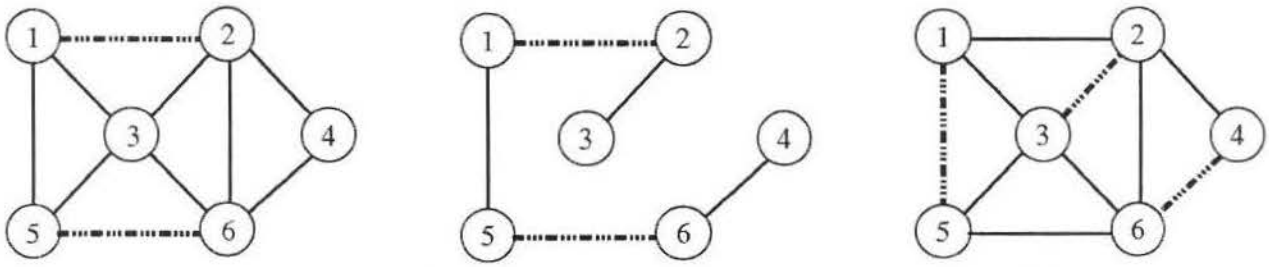


Figura 3.3: *Caminho aumentante para aumentar a cardinalidade de um emparelhamento*

Basicamente, se temos um emparelhamento $\mathcal{M}$ e um caminho aumentante P o emparelhamento resultante $\mathcal{M}^*$, quando definimos as arestas emparelhadas de P como não emparelhadas e vice-versa, possui cardinalidade $|\mathcal{M}^*| = |\mathcal{M}| + 1$ (veja figura 3.3). Denotaremos operação de definir as arestas emparelhadas como não emparelhadas e vice-versa como operação *aumento*. A operação *aumento* aplicada à um ciclo alternante apesar de não aumentar a cardinalidade do emparelhamento será importante no contexto do PD.

O teorema 3.3.2 mostra um resultado análogo ao teorema 3.3.1 para o problema de emparelhamento ponderado. Tarjan [Tar83] apresenta a prova deste teorema.

Teorema 3.3.2 *Seja $\mathcal{M}$ um emparelhamento de custo mínimo dentre todos os emparelhamentos de cardinalidade $|\mathcal{M}|$ e seja P um caminho aumentante para $\mathcal{M}$ de custo mínimo. Denotamos por $\mathcal{M}^* = \mathcal{M} \oplus P$ o emparelhamento resultante de $\mathcal{M}$ aplicando-se a operação* aumento *com o caminho P. O emparelhamento $\mathcal{M}^*$ é o de custo mínimo dentre todos os emparelhamentos de cardinalidade $|\mathcal{M}| + 1$.*

O teorema 3.3.2 implica que o emparelhamento ótimo do PD pode ser obtido de um emparelhamento parcial de custo mínimo, se o emparelhamento é continuamente aumentado a partir de caminhos aumentantes mais curtos.

3.4 Caminhos mais curtos, ciclos negativos e o PD

Um dos problemas de fundamental importância de Fluxos em Redes é o **problema do caminho mais curto**, o qual denotaremos por PCMC. O PCMC é um dos modelos mais simples existente, apesar disso o problema possui muitas características de problemas mais complexos. O PCMC surge frequentemente como um subproblema dos demais problemas de Fluxos em Redes.

Seja $G = (V, E)$ um grafo orientado com custos nas arestas. O PCMC consiste em encontrar o caminho de menor custo entre os vértices de um subconjunto $S \subseteq V$ tal que

$S \neq \emptyset$, e os vértices de um subconjunto $T \subseteq V$ tal que $T \neq \emptyset$. Os subconjuntos S e T não necessariamente são disjuntos. A situação onde $S = T = V$ pode ocorrer.

Quando nos referirmos ao PCMC estaremos tratando do caso particular onde S é unitário e T representa todos os vértices. As possíveis variantes para o problema não possuem a mesma complexidade. Para o PCMC em grafos com arestas de custos não negativos o melhor limite de tempo para a resolução do problema é $O(\min\{m+n\log n,\ m\log\log C,\ m+n\sqrt{\log C}\})$. O limite de tempo polinomial forte é definido pela implementação de Fredman e Tarjan [FT87] do algoritmo de Dijkstra [Dij59]. O limite de tempo polinomial fraco $O(m\log\log C)$ é definido pelo algoritmo de Johnson [Joh82] e o limite $O(m+n\sqrt{\log C})$ pelo algoritmo de Ahuja e outros [AMOT90].

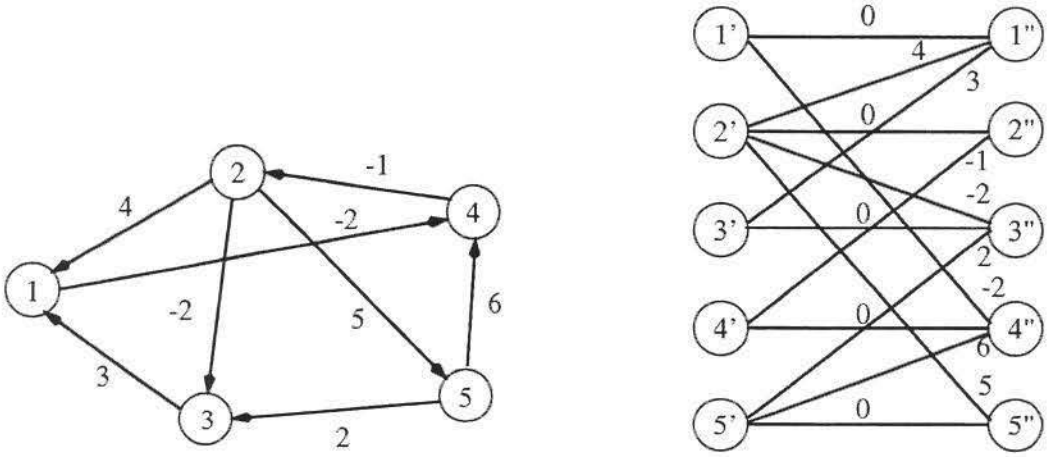
Para o PCMC em grafos com arestas de custos arbitrários supõe-se que o grafo não contém um ciclo de custo negativo, para que os caminhos mais curtos sejam bem definidos. Muitos algoritmos para o PCMC são capazes de detectar a presença de ciclos negativos e por isso não supõem a sua ausência. Para esta variante do problema o melhor limite de tempo para resolução é $O(\min\{nm,\ \sqrt{n}m\log(nC)\})$. O limite de tempo polinomial forte é decorrente do algoritmo de Bellman [Bel58]. O limite de tempo polinomial fraco é devido aos algoritmos de Gabow e Tarjan [GT89a], Orlin e Ahuja [OA92], utilizando a redução do PCMC para o PD.

A diferença entre a complexidade das variantes do PCMC é um dos fatores para a utilização dos potenciais dos vértices (seção 3.1). A manutenção do grafo residual com arestas de custos não negativos possibilita determinar os caminhos mais curtos de maneira mais eficiente. Utilizaremos neste texto $T^{+}_{cmc}(n,m,C)$ para denotar o melhor tempo para a resolução PCMC em grafos com arestas de custos não-negativos e $T_{cmc}(n,m,C)$ para grafos com arestas de custos arbitrários.

O problema do ciclo negativo, o qual denotaremos por PCN, consiste em determinar se um grafo orientado com arestas de custos arbitrários possui um ciclo de custo negativo. Este problema possui também muitas variantes, dentre elas o *problema do ciclo de custo mais negativo* que consiste em determinar o ciclo de custo mais negativo em um grafo, caso exista algum. Este problema ao contrário do primeiro é um problema *NP*-Difícil[6].

O PCN está bastante relacionado ao PCMC com arestas de custos arbitrários. Não somente pelo fato de que a presença de um ciclo negativo implica que não existe uma solução bem definida, mas também porque diversos algoritmos para o PCMC resolvem o PCN. Os melhores limites de tempo polinomial para os dois problemas são fornecidos pelos mesmos algoritmos. O algoritmo de Bellman [Bel58] é o melhor algoritmo de tempo polinomial forte e o algoritmo de Gabow e Tarjan [GT89a], Orlin e Ahuja [OA92] o melhor limite de tempo polinomial fraco. Observe que sobre a suposição de similaridade (seção

[6] O *problema do ciclo hamiltoniano* pode ser reduzido ao *problema do ciclo de custo mais negativo* [GJ79]



ciclo negativo $v_1 - v_4 - v_2 - v_3$ *instância do* PD

Figura 3.4: *Grafo G orientado e o grafo resultante da divisão dos vértices*

1.2), o melhor algoritmo para ambos os problemas é baseado na redução destes ao PD. A redução do PCMC e PCN ao PD será descrita a seguir:

Redução do PCMC e PCN ao PD [7] Seja $G = (V, E)$ um grafo orientado com arestas de custos arbitrários. O objetivo é determinar o menor caminho entre os vértices s e t. Para obtermos uma instância G' do PD, primeiramente realizamos uma operação de divisão dos vértices de G. Cada vértice $v \in V$ será substituído por dois vértices v' e v''. Os vértices v' e v'' serão conectados por uma aresta dirigida (v', v'') de custo zero. Para cada aresta $(v, w) \in E$ insira uma aresta (v', w'') em G' com custo c_{vw} (veja figura 3.4).

Esta transformação gera uma estrutura de grafo bipartido. Antes de determinarmos o menor caminho entre s e t devemos determinar se não existe um ciclo de custo negativo em G.

Pode-se provar que para cada caminho $P = \{v_i, v_{i+1}, \ldots, v_{j-1}, v_j\}$ em G, pode-se obter um emparelhamento $\mathcal{M}$ em G' da forma $\mathcal{M} = \{(v'_i, v''_{i+1}), (v'_{i+1}, v''_{i+2}), \ldots, (v'_{j-2}, v''_{j-1}), (v'_{j-1}, v''_j)\}$ e vice-versa. A mesma afirmação é válida para os ciclos de G. O emparelhamento ótimo em G' pode ser transformado em uma coleção de ciclos orientados disjuntos em G. Para isto substitua cada aresta do emparelhamento (v', w'') por uma aresta dirigida (v, w).

Se o custo do emparelhamento ótimo em G' é negativo pelo menos um dos ciclos orientados correspondentes em G tem custo negativo.

Se o custo do emparelhamento ótimo de G' for zero então não existem ciclos negativos em G. Neste caso, o caminho mais curto entre os vértices s e t pode ser obtido a partir de outra instância $\mathcal{G}$ do PD. Esta nova instância é derivada de G' excluindo os vértices

[7] A redução apresentada é baseada no texto de Ahuja e outros [AMO93]

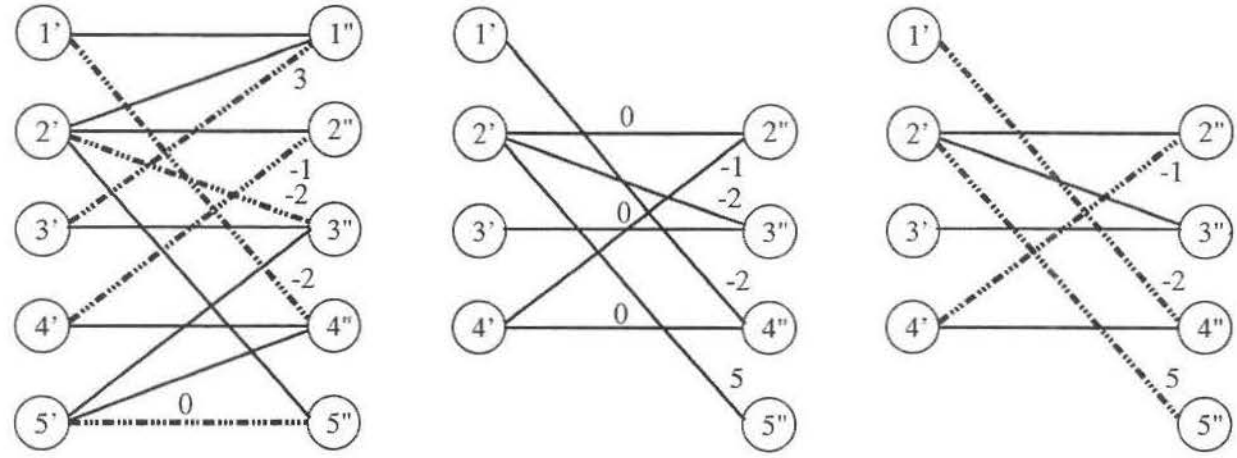


Figura 3.5: *Determinação dos caminhos mais curtos do grafo*

s'' e t'. O emparelhamento ótimo nesta nova instância forma um caminho do vértice s ao vértice t'' mais uma coleção de ciclos orientados disjuntos. Devido G' não conter ciclos negativos, os caminhos mais curtos em G são bem definidos e todos os ciclos possuem custo zero. Desta maneira, o caminho entre o vértice s' e t'' em $\mathcal{G}$ tem que ser o menor caminho entre s e t em G.

Considere a instância $\mathcal{G}$ mostrada na figura 3.5. A aresta (v_2, v_3^*) teve o seu custo alterado de -2 para 1. Desta maneira, não existem ciclos de custo negativo em $\mathcal{G}$ e o caminho mais curto entre os vértices $s = 1$ e $t = 5$ é bem definido.

A maioria dos algoritmos para o PD envolve a determinação do menor caminho aumentante entre os vértices do grafo residual ou a determinação de possíveis ciclos de custo negativo. Por outro lado, o PCMC com custos arbitrários e o PCN podem ser reduzidos ao PD. Os melhores algoritmos de tempo polinomial fraco para o PCMC e PCN são obtidos através desta transformação. Isto demonstra como estes problemas estão fortemente relacionados. A compreensão deste relacionamento é fundamental para resolver variantes do problema de emparelhamento eficientemente.

3.5 Condições de Otimalidade

Pode-se caracterizar a otimalidade de um fluxo ou emparelhamento através de algumas condições que devem ser satisfeitas por qualquer solução ótima para o problema. Essas condições, as quais denominaremos **condições de otimalidade**, são de fundamental importância no entendimento do problema.

As condições de otimalidade frequentemente fornecem algoritmos fáceis e intuitivos para a resolução dos problemas. Além de possibilitar caracterizar a solução ótima, tais condições também indicam em que sentido as soluções intermediárias deveriam ser alteradas para se obter a otimalidade após um número finito de operações.

As condições de otimalidade podem ser consideradas instrumentos eficientes para se estabelecer a corretude dos algoritmos propostos. Se as soluções geradas por estes algoritmos satisfazem uma das condições de otimalidade então estes algoritmos necessariamente resolvem corretamente o problema. Diversos algoritmos propostos para o PD foram baseados nas condições de otimalidade discutidas nesta seção.

As condições de otimalidade empregadas no PD são as condições existentes para o modelo mais genérico, o PFCM. As 3 condições descritas a seguir são equivalentes. A razão por descrever mais de uma é consequência do fato de que nem todos os algoritmos estudados se baseiam na mesma condição. Em alguns casos, uma condição é mais apropriada que as demais por ser de certo modo mais intuitiva[8].

Nesta seção, descrevemos as condições de otimalidade no contexto do PFCM e do PD. As 2 primeiras condições são baseadas no grafo residual e a última condição no grafo original. A descrição das condições de otimalidade apresentada está baseada na demonstração formal encontrada no texto de Ahuja e outros [AMO93].

3.5.1 Condição de otimalidade de Ciclo Negativo

Seja $G(\mathcal{F})$ o grafo residual relativo ao fluxo $\mathcal{F}$ factível. Se $G(\mathcal{F})$ possui um ciclo de custo negativo $\mathcal{C}$, temos que $\mathcal{F}$ não é um fluxo ótimo. Um fluxo de custo menor pode ser obtido a partir do ciclo de custo negativo.

Considere o fluxo obtido de $\mathcal{F}$ aumentando-se o fluxo através do ciclo $\mathcal{C}$. O balanceamento dos vértices permanece inalterado. O custo do novo fluxo, entretanto, é inferior ao custo do fluxo anterior. A cada unidade de fluxo enviada através de $\mathcal{C}$ o custo do fluxo se reduz em $\sum_{(i,j)\in\mathcal{C}} c(i,j)$. Desta forma, o novo fluxo computado será melhor que o anterior.

No caso de um emparelhamento, o aumento de fluxo através do ciclo negativo é equivalente à aplicação da operação *aumento* sobre o ciclo (seção 3.3). O emparelhamento resultante será perfeito e com custo menor visto que o ciclo $\mathcal{C}$ tem custo negativo.

Condição de Otimalidade de Ciclo Negativo *Um fluxo ou emparelhamento perfeito é uma solução ótima se somente se o grafo residual associado não contém nenhum ciclo de custo negativo.*

[8] Uma descrição da equivalência entre estas 3 condições de otimalidade está presente no texto de Ahuja e outros [AMO93]

3.5.2 Condição de otimalidade de Custo Reduzido

Se o grafo residual não contém ciclos de custo negativo, os caminhos mais curtos entre os vértices são bem definidos. Pode-se assim buscar uma interpretação diferente da condição de otimalidade de Ciclo Negativo.

Uma condição necessária simples para se estabelecer a otimalidade dos caminhos mais curtos em um grafo é declarada a seguir:

Lema 3.5.1 *Para cada vértice $v \in V$, denotamos por $d_s(v)$ o tamanho de algum caminho orientado do vértice s ao vértice v. As distâncias $d_s(v)$ representam as distâncias dos caminhos mais curtos se e somente se elas satisfazem a seguinte condição:*

$$d_s(w) \leq d_s(v) + c_{vw} \quad \forall (v,w) \in G$$

O lema 3.5.1 pode ser equivalentemente declarado como $c_{vw} + d_s(v) - d_s(w) \geq 0$. Observe a similaridade com a definição dos custos reduzidos $c^{\pi}_{vw} = c_{vw} - \pi(v) + \pi(w)$. Se definimos o potencial do vértice v como sendo $\pi(v) = -d_s(v)$ temos que $\forall (v,w)\ c^{\pi}_{vw} \geq 0$. Em particular, pode-se provar que para as arestas no menor caminho P entre o vértice s e outro vértice temos $c^{\pi}_e = 0 \quad \forall e \in P$.

Se os caminhos mais curtos no grafo residual são bem definidos então existem potenciais para os quais todas as arestas possuem custo reduzido não negativo. Uma condição de otimalidade decorrente deste teorema é declarada a seguir:

Condição de Otimalidade de Custo Reduzido Um fluxo ou um emparelhamento perfeito é uma solução ótima se somente se existe um conjunto de potenciais π que satisfaz a condição:

$$c^{\pi}_e \geq 0 \qquad \forall e \text{ no grafo residual}$$

Os potenciais para o qual a condição de otimalidade de Custo Reduzido é satisfeita, podem ser obtidos determinando o menor caminho de um vértice qualquer aos demais vértices do grafo residual.

Os caminhos mais curtos no grafo residual em relação aos custos originais ou aos custos reduzidos são os mesmos, embora os custos dos caminhos possam diferir. Tomizawa [Tom72] e Edmonds e Karp [EK72] independentemente observaram que se os algoritmos utilizam os potenciais é possível manter os custos das arestas não-negativo. Desta maneira, os caminhos mais curtos podem ser calculados mais eficientemente se utilizamos os custos reduzidos das arestas. Um dos principais benefícios na utilização dos potenciais vêm dessa observação. Os caminhos mais curtos são computações realizadas frequentemente por inúmeros algoritmos para os mais diversos problemas de Fluxos em Redes. A possibilidade de trabalhar com grafos com arestas de custos não-negativos tem influência imediata nestes algoritmos.

3.5.3 Condição de otimalidade de Folga Complementar

As condições de otimalidade descritas anteriormente são definidas em relação ao grafo residual. A *condição de otimalidade de Folga Complementar* é definida em relação ao grafo original.

Se um fluxo é ótimo em relação ao conjunto de potenciais π, as arestas no grafo original com custos reduzidos positivos devem ter fluxo nulo. Caso contrário, as arestas reversas estarão presentes no grafo residual com custos reduzidos negativos em relação a π. Da mesma maneira, as arestas com custos reduzidos negativos devem estar saturadas para que tenham capacidade residual nula e não estejam no grafo residual. Por outro lado, não existem restrições quanto ao fluxo presente nas arestas com custo reduzido nulo. Em qualquer situação estas arestas e as suas reversas, caso estejam no grafo residual, satisfarão a condição de otimalidade de Custo Reduzido.

A partir destas observações pode-se formular condições para avaliar a otimalidade de um fluxo a partir do grafo original. Estas condições são descritas a seguir:

Condição de Otimalidade de Folga Complementar Um fluxo $\mathcal{F}$ é uma solução ótima do PFCM se e somente se para algum conjunto de potenciais π, os custos reduzidos satisfazem a seguinte condição de otimalidade para cada aresta $e \in E$:

$$\begin{aligned} &se\ c^{\pi}_{vw} > 0 \quad então\ x_{vw} = 0 \\ &se\ c^{\pi}_{vw} < 0 \quad então\ x_{vw} = u_{vw} \\ &se\ c^{\pi}_{vw} = 0 \quad então\ l_{vw} \leq x_{vw} \leq u_{vw} \end{aligned}$$

Em relação à um emparelhamento, somente arestas saturadas e de fluxo nulo estão presentes. A condição anterior pode ser especializada para o PD como descrito a seguir:

Condição de Otimalidade de Folga Complementar para o PD Um emparelhamento perfeito $\mathcal{M}$ é uma solução ótima do PD se somente se para algum conjunto de potenciais π, os custos reduzidos satisfazem a seguinte condição de otimalidade para cada aresta $e \in E$:

$$\begin{aligned} &se\ e \notin \mathcal{M} \quad então\ c^{\pi}_{e} \geq 0 \\ &se\ e \in \mathcal{M} \quad então\ c^{\pi}_{e} \leq 0 \end{aligned}$$

Capítulo 4

Algoritmos

Neste capítulo discutimos as principais abordagens utilizadas na resolução do PD e do PFCM. O texto abrange apenas uma parcela dos algoritmos aplicados aos problemas. Ahuja e outros [AMO93] e Bertsekas [Ber91] discutem mais detalhadamente a variedade de algoritmos existentes.

A maioria dos algoritmos especializados para o PD são algoritmos para um problema mais genérico, como por exemplo o PFCM, que exploram as particularidades da estrutura do problema. Exceções a esta regra existem. Um dos maiores exemplos é o clássico algoritmo *primal-dual* de Kuhn [Kuh55, Kuh56] para o PD. Este algoritmo desenvolvido especificamente para o PD foi generalizado por Ford e Fulkerson [FF57, FF62] dando origem ao algoritmo *primal-dual* para o PFCM.

Cada um dos algoritmos apresentados para o PD e PFCM se fundamenta em uma das condições de otimalidade descritas na seção 3.5. Pode-se visualizar três abordagens específicas utilizadas pelos diversos algoritmos para o problema:

1. Ciclos negativos;
2. Caminhos aumentantes;
3. Otimalidade aproximada.

Nas subseções seguintes discutimos os aspectos mais importantes destas abordagens.

4.1 Ciclos negativos

Os algoritmos com esta abordagem utilizam a condição de otimalidade de Ciclo Negativo para obter uma solução ótima a partir de uma solução inicial factível. Estes algoritmos detectam e cancelam os ciclos de custo negativo presentes no grafo residual. Os algoritmos

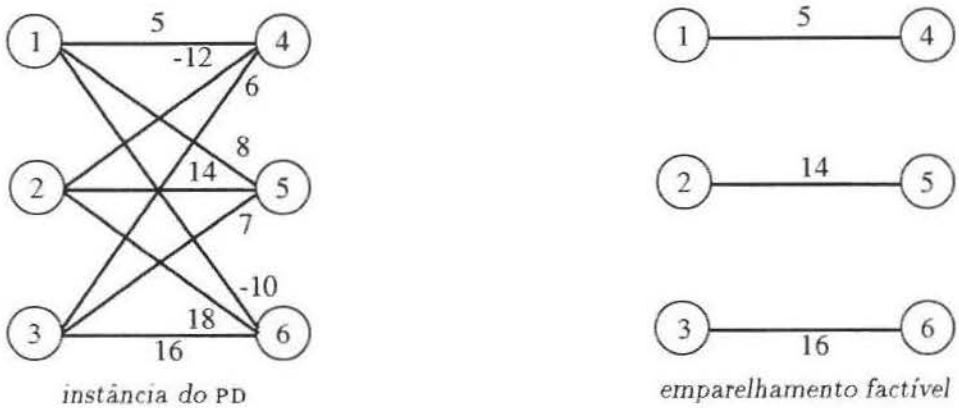


Figura 4.1: *Instância utilizada para a explicação do comportamento dos algoritmos*

terminam quando a condição de otimalidade de Ciclo Negativo é satisfeita. A solução inicial mencionada pode ser obtida determinando o fluxo máximo no grafo original.

Para cancelar um ciclo de custo negativo C, o algoritmo aumenta o fluxo o máximo possível através do ciclo. No mínimo uma aresta no ciclo se saturará, sendo excluída do grafo residual. Desta forma, o grafo residual do fluxo resultante não contém o ciclo C. No contexto do PD o cancelamento do ciclo pode ser interpretado como a aplicação da operação *aumento* (seção 3.3). No grafo residual do emparelhamento resultante, todas as arestas do ciclo inverteram suas orientações e desta forma o ciclo C possui custo não-negativo.

Para explicar brevemente a execução dos algoritmos estudados utilizaremos a mesma instância do PD como exemplo. A aplicação dos algoritmos para instâncias do PFCM é similar. Na ilustração 4.1 é especificado a instância utilizada e um emparelhamento factível. Na figura 4.2 exibimos o comportamento comum dos algoritmos baseados em cancelamento de ciclos. Um ciclo de custo negativo está presente no grafo residual do emparelhamento factível, o que indica que este emparelhamento não é ótimo. A aplicação da operação *aumento* sobre este ciclo resulta no grafo residual presente na mesma figura. O emparelhamento correspondente à este grafo residual é ótimo, já que não contém ciclos negativos.

4.1.1 Cancelamento de ciclos

O algoritmo de **cancelamento de ciclos** foi proposto por Klein [Kle67]. O algoritmo sucessivamente detecta ciclos de custo negativo no grafo residual e os cancela aumentando o fluxo o máximo possível através do ciclo. Os ciclos podem ser encontrados aplicando-se para isto algum algoritmo capaz de detectar ciclos negativos.

O algoritmo de cancelamento de ciclos genérico não especifica a ordem na qual os ciclos

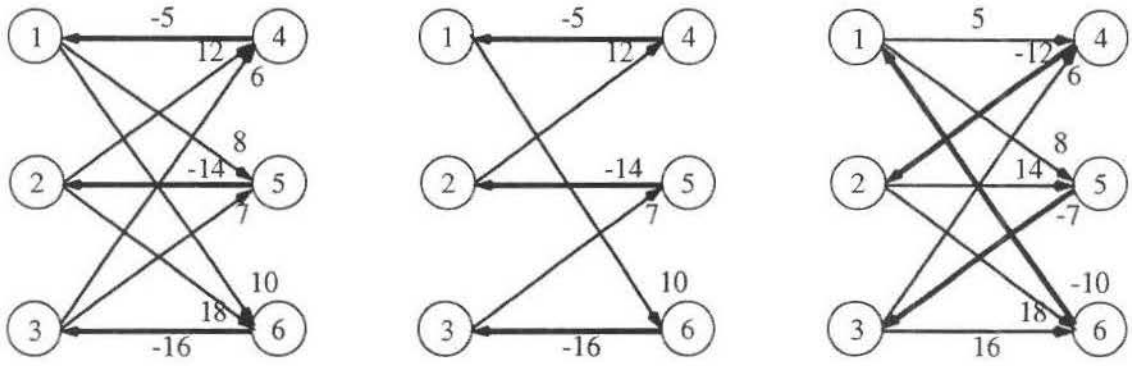


Figura 4.2: *Algoritmo baseado no cancelamento de ciclos negativos*

negativos devem ser cancelados. A implementação do algoritmo nestas condições possui complexidade apenas pseudopolinomial: $O(nm^2C)$ para o PD e $O(nm^2CU)$ para o PFCM. Versões em tempo polinomial do algoritmo existem especificando o tipo de ciclo a ser cancelado. Essas variantes são o algoritmo de *cancelamento de ciclos de média mínima* de Goldberg e Tarjan [GT89b] e o algoritmo de *cancelamento de ciclos de razão mínima* de Wallacher e Zimmerman [WZ91].

O algoritmo de *cancelamento de ciclos de média mínima* sempre cancela um ciclo negativo com média mínima. A média de um ciclo é definida como sendo o seu custo dividido pelo número de arestas no ciclo.

O melhor limite de tempo polinomial para o *problema do ciclo de média mínima*, denotado por PCMM, é $O(\min\{nm, \sqrt{n}m\log(nC)\})$[1]. O limite de tempo polinomial forte é definido pelo algoritmo de Karp [Kar78] utilizando técnicas de Programação Dinâmica. O limite de tempo polinomial fraco é definido pelo algoritmo de Orlin e Ahuja [OA92]. Orlin e Ahuja mostram como o ciclo de média mínima pode ser obtido com um procedimento de busca binária aproximada, aplicando-se um algoritmo para o PD. Observe que o ciclo de média mínima em um grafo pode ser determinado com a mesma complexidade necessária para se determinar um ciclo negativo (seção 3.4).

Se os ciclos cancelados sempre forem os ciclos de média mínima o número de ciclos negativos encontrados é limitado polinomialmente a $O(\min\{nm\log(nC),\ nm^2\log n\})$. Em conseqüência, o algoritmo cancelamento de ciclos de Goldberg e Tarjan [GT89b] tem complexidade polinomial $O(\min\{n^2m^2\log(nC), n^2m^3\log n\})$.

[1] O PCMM aparece na resolução de problemas combinatórios como o algoritmo de circulação de custo mínimo de Goldberg e Tarjan [GT89b] e no algoritmo de balanceamento mínimo de Schneider e Schneider [SS89]

4.1.2 Simplex de rede

O algoritmo **simplex de rede**, proposto por Dantzig [Dan51], é uma especialização do clássico algoritmo *simplex* da Programação Linear para os problemas de Fluxos em Redes. Papadimitriou e Steiglitz [PS98] apresentam um estudo extenso sobre Programação Linear e o algoritmo simplex.

Para explicar os conceitos do algoritmo simplex de rede, classificaremos as arestas de uma solução quanto ao fluxo presente na mesma. Uma aresta que apresenta fluxo nulo ou igual à sua capacidade será denominada *aresta restrita*. As demais arestas que não satisfazem esta condição serão denominadas *arestas irrestritas*. O termo *restrita* está relacionado ao fato de que o fluxo na aresta não pode ser alterado livremente sem violar as restrições quanto ao limite de fluxo nas arestas (não se pode aumentar o fluxo nas arestas saturadas e não se pode reduzir o fluxo nas arestas com fluxo nulo). Em termos do grafo residual, se uma aresta é restrita a aresta reversa correspondente não está presente.

Uma solução para o PFCM é denominada **livre de ciclo** se não existem ciclos no grafo residual onde todas as arestas são irrestritas. Uma importante propriedade do PFCM relacionada às soluções livres de ciclo é declarada a seguir. A prova apresentada deste teorema está baseada na descrição presente no texto de Ahuja e outros [AMO93].

Propriedade livre de ciclo Se uma instância do PFCM possui solução factível então o problema tem sempre uma solução ótima livre de ciclo.

Qualquer solução factível para o PFCM pode ser convertida em uma solução equivalente livre de ciclo. Suponha a existência de um fluxo ótimo $\mathcal{F}$ para o PFCM que não é uma solução livre de ciclo. O motivo pelo qual este fluxo não é livre de ciclo é a presença do ciclo orientado $\mathcal{C}$ contendo somente arestas irrestritas. Se o custo do ciclo $\mathcal{C}$ é não nulo, o fluxo $\mathcal{F}$ não pode ser ótimo. Observe que cada aresta irrestrita está presente no grafo residual assim como à aresta reversa correspondente. Desta forma, se o custo de $\mathcal{C}$ é não nulo o grafo residual possui um ciclo negativo. Pela condição de otimalidade de ciclo negativo este fluxo não pode ser ótimo.

Se $\mathcal{F}$ é um fluxo ótimo então qualquer ciclo $\mathcal{C}$ contendo somente arestas irrestritas possui custo zero. O aumento (ou a redução) do fluxo através de $\mathcal{C}$ não altera a factibilidade da solução e principalmente não altera o seu custo. Pode-se assim obter uma solução factível derivada de $\mathcal{F}$ livre de ciclo aumentando (ou reduzindo) o fluxo através de $\mathcal{C}$ até que uma de suas arestas se torne restrita.

Uma solução para o PFCM é denominada solução de **árvore espalhada** se é possível definir uma árvore espalhada $\mathcal{T}$ (um subgrafo acíclico conectando todos os vértices) para a solução, de forma que todas as arestas não presentes em $\mathcal{T}$ são restritas. Observe que nenhuma imposição foi feita com relação às arestas da árvore. Assim como as soluções

livre de ciclo é sempre possível encontrar uma solução de árvore espalhada ótima para uma instância do PFCM.

A cada solução de árvore espalhada está associada uma única solução livre de ciclo. É possível definir uma solução de árvore espalhada a partir de uma solução livre de ciclo. As arestas irrestritas numa solução livre de ciclo definem uma floresta para a instância. Se esta floresta é uma árvore a solução então é uma solução de árvore espalhada. Caso contrário, pode-se adicionar arestas restritas à floresta até que se obtenha uma árvore. Para a árvore gerada esta solução será uma solução de árvore espalhada. Observe que pela maneira como é obtida, pode haver várias soluções de árvores espalhadas distintas para representar um mesmo fluxo.

Uma solução de árvore espalhada será descrita através de uma estrutura de árvore espalhada $\mathcal{A} = (\mathcal{T}, \mathcal{L}, \mathcal{U})$. A estrutura $\mathcal{A}$ particiona as arestas do grafo. A partição $\mathcal{T}$ são as arestas da árvore espalhada, a partição $\mathcal{L}$ as arestas com fluxo nulo não presentes em $\mathcal{T}$ e a partição $\mathcal{U}$ as arestas saturadas não presentes em $\mathcal{T}$.

Dada uma estrutura de árvore espalhada $\mathcal{A}$, pode-se obter a solução associada à mesma definindo fluxo nulo nas arestas de $\mathcal{L}$, saturando as arestas de $\mathcal{U}$ e resolver as restrições de balanceamento de fluxo nos vértices para determinar o fluxo nas arestas de $\mathcal{T}$. Uma estrutura $\mathcal{A}$ é considerada factível se a solução associada à esta estrutura é factível para o problema. As soluções de árvore espalhada em que todas as arestas de $\mathcal{T}$ são irrestritas serão denominadas *soluções não degeneradas*, caso contrário *degeneradas* .

Se uma solução de árvore espalhada é ótima então esta satisfaz a condição de otimalidade de custo reduzido. Esta condição especializada para a estrutura de árvore espalhada é:

Condição de otimalidade de Custo Reduzido especializada Uma estrutura de árvore espalhada $\mathcal{A} = (\mathcal{T}, \mathcal{L}, \mathcal{U})$ é ótima se para algum conjunto de potenciais π temos que:

$$\begin{aligned} c^{\pi}_{vw} &= 0 && \forall (v,w) \in \mathcal{T} \\ c^{\pi}_{vw} &> 0 && \forall (v,w) \in \mathcal{L} \\ c^{\pi}_{vw} &< 0 && \forall (v,w) \in \mathcal{U} \end{aligned}$$

O algoritmo simplex de rede inicia com uma solução de árvore espalhada obtida a partir de algum fluxo factível para o problema. Se este fluxo não é ótimo então no mínimo uma das arestas não presente na árvore espalhada viola a condição de otimalidade. O algoritmo escolhe uma das arestas que viola a condição, utilizando um critério conveniente, e adiciona a aresta na árvore espalhada. A inserção da aresta na árvore cria um ciclo negativo. O algoritmo aumenta o fluxo através do ciclo o máximo possível para cancelá-lo. A aresta do ciclo que se torna restrita é retirada obtendo-se assim uma nova estrutura

de árvore espalhada. A passagem de uma árvore espalhada para outra é denominada *pivotamento* devido à relação do algoritmo com a Programação Linear. O algoritmo prossegue realizando pivotamentos até a obtenção de uma solução ótima.

O desempenho do algoritmo simplex de rede depende de como as informações sobre a estrutura de árvore espalhada e como os pivotamentos são implementados. Os vários critérios utilizados para a escolha da aresta para o pivotamento geram variantes do algoritmo. Cada uma dessas variantes possui um desempenho específico.

O algoritmo simplex de rede pode não convergir finitamente para o fluxo ótimo se precauções não são tomadas quanto à aresta retirada num pivotamento. O problema ocorre por causa da degenerescência, ou seja, a obtenção de soluções degeneradas. Nas soluções degeneradas a inserção de uma nova aresta não gera ciclos orientados. Se a inserção da aresta não resulta em um ciclo o fluxo não será modificado. Neste caso, o algoritmo obterá uma estrutura de árvore espalhada equivalente para a mesma solução.

Uma implementação em tempo polinomial para o algoritmo simplex de rede somente recentemente foi proposta por Orlin [Orl97] com complexidade $O(\min\{n^2 m \log(nC), n^2 m^2 \log n\})$. Implementações em tempo polinomial para os casos específicos do PFCM já haviam sido propostas anteriormente. Orlin e Ahuja [OA92] obtiveram uma implementação do algoritmo simplex de rede para o PD com tempo polinomial $O(nm \log C)$.

4.2 Caminhos aumentantes mais curtos

Os algoritmos baseados nesta abordagem iniciam com soluções que satisfazem a condição de otimalidade de Custo Reduzido mas não necessariamente são soluções factíveis.

O algoritmo obtem sucessivamente novas soluções mantendo-se a condição de otimalidade satisfeita. A cada iteração o algoritmo diminui o grau de infactibilidade das soluções, até a obtenção de uma solução que seja ótima e factível. No caso do PD cada solução obtida é um emparelhamento parcial ótimo, emparelhamento de custo mínimo para um emparelhamento com cardinalidade $k < n$.

A solução ótima inicial para o PFCM pode ser, por exemplo, um pseudofluxo nulo. No caso do PD a solução inicial é um emparelhamento de cardinalidade nula. Observe que a instância do problema satisfaz à condição de otimalidade de Custo Reduzido para $\pi = 0$. Neste caso, o grafo residual é o próprio grafo original. Suponha que o grafo residual associado à esta solução possui um caminho aumentante que seja o menor caminho entre os vértices v e w. O aumento do fluxo através deste caminho reduz o desbalanceamento nos vértices v e w. Por ser o menor caminho entre dois vértices, todas as arestas do caminho possuem custo reduzido nulo de maneira que o aumento do fluxo através deste caminho não viola a condição de otimalidade. O fluxo resultante é melhor do que o anterior no sentido de que a infactibilidade foi reduzida.

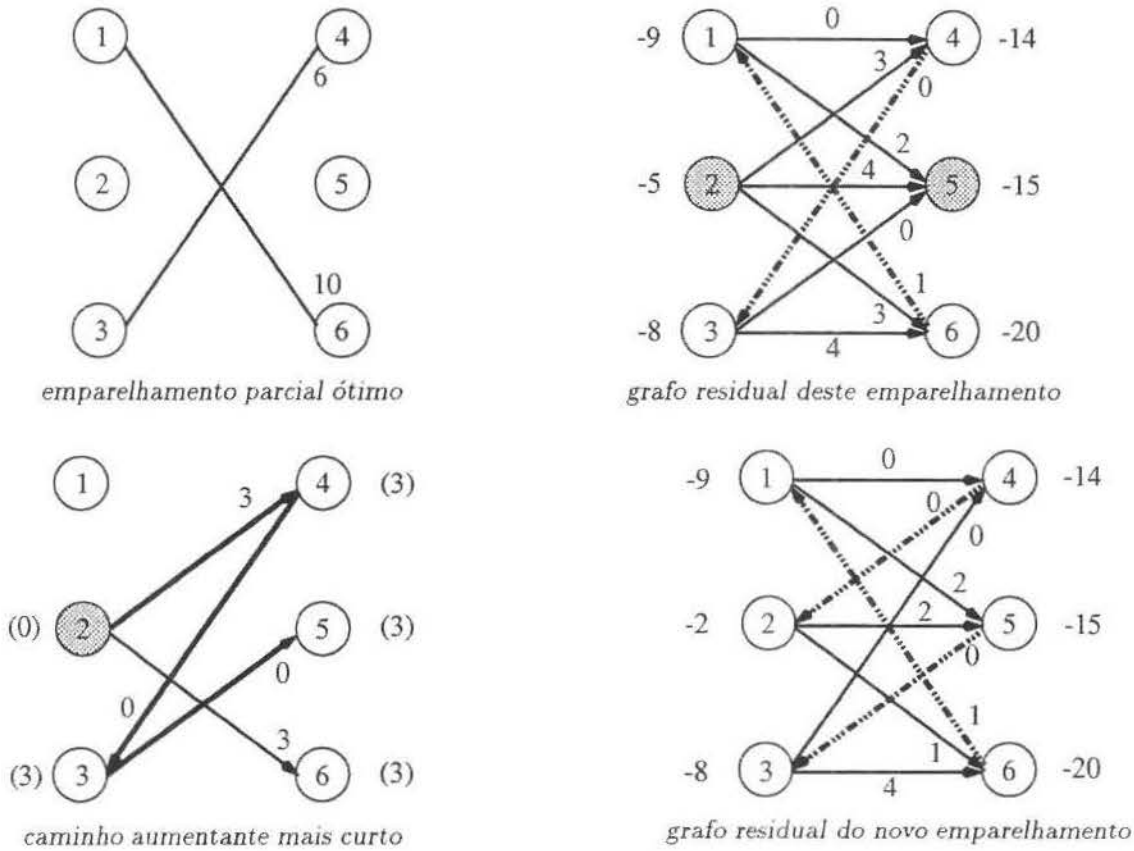


Figura 4.3: *Comportamento dos algoritmos baseado em caminhos aumentantes*

Na ilustração 4.3 é mostrada a iteração básica dos algoritmos baseados em caminhos aumentantes. Observe que o grafo residual do emparelhamento parcial satisfaz a condição de otimalidade de Custo Reduzido para os potenciais mostrados. Neste emparelhamento os vértices v_2 e v_5 são vértices não emparelhados. Um dos caminhos na árvore de caminhos mais curtos ilustrada é um caminho aumentante (caminho $v_2 - v_4 - v_3 - v_5$). Os valores associados à cada vértice na figura são as distâncias na árvore. O emparelhamento resultante satisfaz a condição de otimalidade de Custo Reduzido para os potenciais mostrados. Concluímos então que este emparelhamento é o emparelhamento ótimo procurado.

4.2.1 Caminhos mais curtos sucessivos

O algoritmo de **caminhos mais curtos sucessivos** computa sucessivos pseudofluxos ótimos até a obtenção de um fluxo ótimo. Este algoritmo inicia com um pseudofluxo inicial ótimo, geralmente um pseudofluxo nulo. A cada iteração o algoritmo reduz o desbalanceamento nos vértices desta solução, mantendo-se a otimalidade, através dos caminhos aumentantes mais curtos.

A árvore de caminhos mais curtos não precisa ser computada completamente. O algoritmo pode terminar assim que o menor caminho aumentante for encontrado. Suponha que o caminho aumentante P encontrado é orientado do vértice v ao vértice w. Os potenciais devem ser atualizados da seguinte maneira:

$$\pi(u) = \begin{cases} \pi(u) - d_v(u) & \text{se existe um caminho na árvore alternante de } v \text{ a } u \\ \pi(u) - d_v(w) & \text{caso contrário} \end{cases}$$

Outra forma de atualização é a seguinte:

$$\pi(u) = \begin{cases} \pi(u) - d_v(u) + d_v(w) & \text{se existe um caminho na árvore alternante de } v \text{ a } u \\ \pi(u) & \text{caso contrário} \end{cases}$$

É semelhante à anterior com o acréscimo de $d_v(w)$. A vantagem desta nova forma de atualização é que os potenciais dos vértices não presentes na árvore de caminhos mais curtos não necessitam ser atualizados. Em seguida o algoritmo obtém um novo pseudofluxo enviando o máximo possível de fluxo através do caminho P.

A cada iteração o algoritmo reduz o desbalanceamento dos vértices. Se denotamos por D o maior excesso ou demanda de fluxo em um vértice temos que o algoritmo de caminhos mais curtos sucessivos realiza no máximo nD iterações. A complexidade do algoritmo de caminhos mais curtos sucessivos para o PFCM é $O(nD\, T^+_{CMC}(n, m, C))$.

No caso do PD a árvore de caminhos mais curtos tem como origem um dos vértices não emparelhados. A cada iteração a cardinalidade do emparelhamento aumenta uma unidade. Após n iterações o algoritmo determina um emparelhamento perfeito ótimo. A complexidade deste algoritmo para o PD é $O(n\, T^+_{CMC}(n, m, C))$.

4.2.2 Primal-dual

O clássico algoritmo **primal-dual** proposto por Kuhn [Kuh55, Kuh56] foi o primeiro algoritmo específico para o PD[2].

O algoritmo primal-dual, assim como o algoritmo de caminhos mais curtos sucessivos inicia com um pseudofluxo ótimo. Em seguida, sucessivos pseudofluxos são gerados pelo algoritmo, mantendo-se a otimalidade, até a obtenção de um fluxo ótimo. O algoritmo primal-dual se diferencia por buscar melhorar o pseudofluxo, se possível, através de vários caminhos aumentantes simultaneamente.

O objetivo do algoritmo é obter caminhos aumentantes disjuntos no grafo residual e reduzir o desbalanceamento a cada iteração no maior número possível de vértices. Os caminhos aumentantes podem ser obtidos simplesmente construindo uma árvore de caminhos mais curtos para cada vértice não balanceado. Determina-se então a existência

[2]Kuhn denominou o algoritmo como **método húngaro** em reconhecimento aos trabalhos de König [Kön31, Kön50] e Egérvary [Egé31] para a teoria de emparelhamentos

de possíveis caminhos aumentantes disjuntos nestas árvores. Para obter os caminhos de modo mais eficiente, o algoritmo resolve uma instância do PFM em um grafo contendo apenas as arestas de custo reduzido zero.

A resolução do PFM equivale ao envio de fluxo através de vários caminhos aumentantes do grafo residual simultaneamente. O número máximo de iterações no algoritmo para o PFCM é limitado a $O(\min\{nU, nC\})$. A complexidade de cada iteração é definida pela complexidade da computação dos caminhos mais curtos e do fluxo máximo $O(\min\{nU, nC\}\{T^{+}_{CMC}(n, m, C) + T_{FM}(n, m, U)\})$.

A complexidade do algoritmo primal-dual para o PD é $O(nT^{+}_{CMC}(n, m, C))$. A computação dos fluxos máximos é dominada pelo tempo necessário para a computação da árvore de caminhos mais curtos. A primeira implementação do algoritmo possuía complexidade $O(n^4)$. Posteriormente implementações mais eficientes foram propostas utilizando-se os custos reduzidos e potenciais. O melhor limite de tempo polinomial forte para o PD $O(nm + n^2 \log n))$ é obtido com a implementação do algoritmo primal-dual utilizando o algoritmo de Dijkstra [Dij59] para o PCMC com a estrutura de lista de prioridades de Fibonacci de Fredman e Tarjan [FT87].

4.3 Otimalidade aproximada

O conceito de **otimalidade aproximada** foi desenvolvido independentemente por Bertsekas [Ber79] e Tardos [Tar85]. Este conceito, descrito a seguir, consiste na relaxação da condição de otimalidade de Folga Complementar.

Condição de otimalidade de Folga ϵ Complementar Um fluxo $\mathcal{F}$ é considerado ótimo para algum $\epsilon > 0$ se existe um conjunto de potenciais π para o qual a seguinte condição de otimalidade de Folga ϵ Complementar é satisfeita

$$\begin{array}{ll} \text{se } c^{\pi}_{vw} > \epsilon & \text{então } x_{vw} = 0 \\ \text{se } -\epsilon \leq c^{\pi}_{vw} \leq \epsilon & \text{então } l_{vw} \leq x_{vw} \leq u_{vw} \\ \text{se } c^{\pi}_{vw} < -\epsilon & \text{então } x_{vw} = u_{vw} \end{array}$$

A condição de otimalidade de Folga ϵ Complementar se reduz à condição de Folga Complementar original quando $\epsilon = 0$. Esta condição pode ser declarada de forma equivalente em relação ao grafo residual como:

Condição equivalente Um fluxo $\mathcal{F}$ é considerado ótimo para algum $\epsilon > 0$ se existe um conjunto de potenciais π para o qual o grafo residual $G(\mathcal{F})$ satisfaz a seguinte condição

$$c^{\pi}_{e} \geq -\epsilon \qquad \forall e \in G(\mathcal{F})$$

Um fluxo que satisfaz à condição de otimalidade aproximada está $n\epsilon$ da otimalidade, ou seja, a diferença entre o custo do fluxo corrente e do fluxo ótimo não pode ser superior a $n\epsilon$. Um fluxo que é ótimo para ϵ tem de ser ótimo se os custos das arestas são inteiros e $n\epsilon < 1$. Este resultado é apresentado na proposição 4.3.1 declarada a seguir. A prova desta proposição está descrita nos textos de Ahuja e outros [AMO93], Bertsekas [Ber91] e Bertsekas e Tsitsiklis [BT89].

Proposição 4.3.1 *Algum fluxo é ótimo para ϵ quando $\epsilon \geq C$. Se os custos das arestas são inteiros, um fluxo satisfazendo à condição de otimalidade de Folga ϵ Complementar para algum $\epsilon < \frac{1}{n}$ é um fluxo ótimo.*

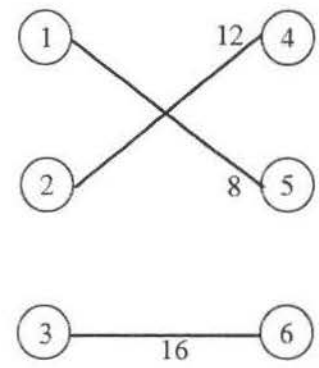
O principal motivo para se utilizar a condição aproximada é que para valores suficientemente pequenos de ϵ, um fluxo satisfazendo a condição aproximada é ótimo. Desta forma, não é necessário resolver o problema até a condição exata ser satisfeita, para pequenos valores de ϵ aproximações são suficientes.

Os algoritmos baseados no conceito de otimalidade aproximada usam um valor de $\epsilon > 0$ fixo e iniciam com um pseudofluxo satisfazendo a condição de otimalidade de Folga ϵ Complementar. A cada iteração o algoritmo escolhe um vértice com excesso e tenta reduzir o seu excesso através da melhor aresta incidente ao vértice. A melhor aresta depende fundamentalmente das características do algoritmo. O algoritmo preserva a otimalidade em todas as iterações até a obtenção de um fluxo. Conforme mostrado acima, se ϵ é suficientemente pequeno então o fluxo é ótimo.

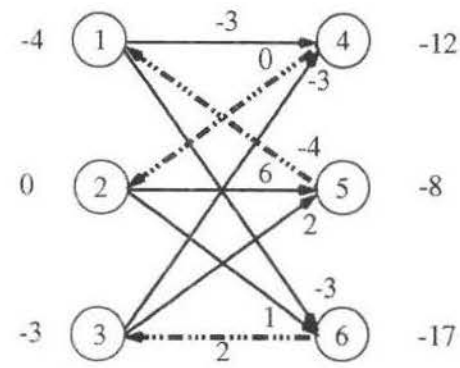
Os algoritmos baseados no conceito de otimalidade aproximada são adequados à utilização da técnica de aproximação por *escalonamento* criada por Edmonds e Karp [EK72]. **Escalonamento** consiste em se aplicar o algoritmo diversas vezes iniciando com uma aproximação fraca. A aproximação é refinada continuamente até se obter uma solução que se possa garantir ser ótima. Cada aplicação do algoritmo freqüentemente provê boas aproximações iniciais para a próxima aplicação.

Escalonamento explora o fato de que para alguns problemas, obter uma solução ótima com uma boa aproximação inicial é mais simples do que resolver o problema. A primeira aproximação pode ser suficientemente fraca, o quanto for necessário. Em geral, obter uma solução inicial para esta aproximação é trivial. Em seguida, a aproximação deve ser melhorada por um fator α constante.

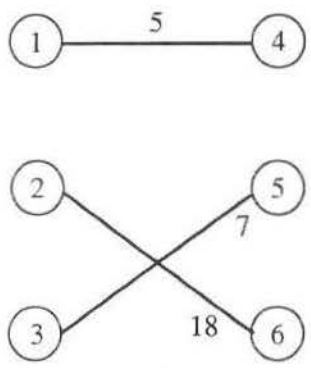
Nas ilustração 4.4 são mostrados os conceitos básicos das versões escalonadas dos algoritmos baseados em otimalidade aproximada. Na figura é exibido um emparelhamento perfeito satisfazendo a condição de otimalidade relaxada para $\epsilon = 4$, o grafo residual e os potenciais para o qual o mesmo é ótimo para ϵ. Melhorando a aproximação inicial por um fator $\alpha = 2$, obtêm-se o emparelhamento seguinte o qual é ótimo para $\epsilon = 2$. A maneira como a aproximação inicial é melhorada depende fundamentalmente das características do algoritmo.



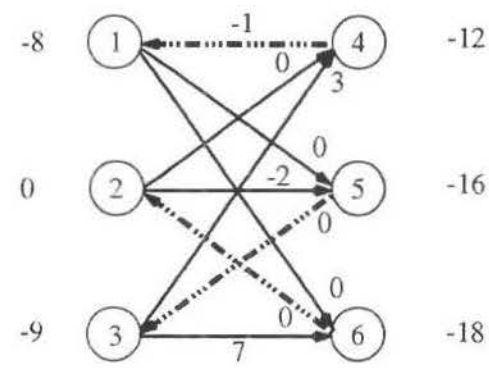
emparelhamento ótimo para $\epsilon = 4$



Condição de otimalidade relaxada satisfeita



emparelhamento ótimo para $\epsilon = 2$



novo emparelhamento mais próximo da otimalidade

Figura 4.4: *Comportamento dos algoritmos baseados em otimalidade aproximada*

4.3.1 Escalonamento de custos

O algoritmo de **escalonamento de custos** utiliza os conceitos de otimalidade aproximada e escalonamento descritos, mais os conceitos do *algoritmo pré-fluxo* de Goldberg e Tarjan [GT88] para o PFM.

Algoritmo pré-fluxo No algoritmo pré-fluxo os vértices com excesso de fluxo são denominados *ativos*. Um fluxo é um pseudofluxo onde não existem vértices ativos. Por definição, a fonte e o sorvedouro jamais se tornam ativos. A cada vértice v é associado um *rótulo* $d(v)$ que varia dentro da faixa de valores $[0, 2n-1]$. Se as condições $d(t) = 0$, $d(s) = n$ e $d(v) \leq d(w) + 1$ são satisfeitas para cada aresta no grafo residual (v, w) os rótulos são denominados *válidos*.

O **algoritmo pré-fluxo** consiste em ao invés de enviar fluxo da fonte para o sorvedouro, através de um caminho aumentante, decompor esta operação em várias operações mais simples. Nas operações básicas do algoritmo, se envia o fluxo de um vértice ativo para um vértice adjacente que se estima esteja mais próximo do sorvedouro. Pode-se

estimar a proximidade entre um vértice e o sorvedouro (ou a fonte) através do seu rótulo. O algoritmo termina quando não existem mais vértices ativos.

As operações básicas do algoritmo são as operações `enviar/renomear` (`push/relabel`):

`enviar` consiste em enviar o excesso de fluxo para o vértice adjacente que se estima esteja mais próximo do sorvedouro. Caso não seja possível alcançar o sorvedouro, o fluxo é enviado para o vértice adjacente mais próximo da fonte.

`renomear` se todos os vizinhos no grafo residual tem rótulo igual ou maior, atualiza-se o rótulo deste vértice aumentando-o o mínimo possível. Durante a execução do algoritmo o rótulo de um vértice jamais decresce.

O algoritmo de escalonamento de custos pode ser visualizado como uma generalização do algoritmo pré-fluxo aplicada ao PFCM. O algoritmo de escalonamento de custos inicia com um pseudofluxo nulo ótimo para $\epsilon = C$. Sucessivamente o algoritmo reduz o valor de ϵ por um fator constante $\alpha > 0$ e obtém fluxos ótimos para $\frac{\epsilon}{\alpha}$. O algoritmo termina após $1 + \lfloor \log_\alpha(nC) \rfloor$ iterações escalonadas quando um fluxo ótimo para $\epsilon < \frac{1}{n}$ é determinado.

A principal iteração do algoritmo consiste em se obter um fluxo ótimo para $\frac{\epsilon}{\alpha}$ a partir de um fluxo ótimo para ϵ. O algoritmo inicialmente define um pseudofluxo ótimo para $\frac{\epsilon}{\alpha}$, como por exemplo, um pseudofluxo nulo. O algoritmo então converte o pseudofluxo em um fluxo ótimo para $\frac{\epsilon}{\alpha}$. O algoritmo busca obter o fluxo através de operações simples sobre os vértices ativos e sobre as suas arestas admissíveis. As arestas *admissíveis* de um vértice para um determinado fluxo são as arestas de custo reduzido negativo. As operações `enviar/renomear` são redefinidas a seguir:

`enviar` é aplicada a um vértice ativo v. Consiste em enviar o máximo possível de fluxo δ através de uma de suas arestas admissíveis (v, w),

$$\delta = \min\{D(v), r(v, w)\}$$

`renomear` aplica-se a um vértice ativo v que não possui arestas admissíveis. Consiste em reduzir o potencial do vértice o mais possível sem violar a condição de otimalidade

$$\pi(v) = \max_{(v,w)\in G(\mathcal{F})}\{\pi(w) - c(v, w) - \epsilon\}$$

A versão genérica do algoritmo não menciona a ordem na qual os vértices ativos devem ser processados. Para diferentes ordens de execução o algoritmo apresenta a mesma complexidade, entretanto, o desempenho prático do algoritmo pode variar consideravelmente. Em uma fase escalonada o número máximo de operações `renomear` é limitado a $O(n^2)$ e o número de operações `enviar` a $O(n^2m)$. A implementação do algoritmo utilizando

árvores dinâmicas de Goldberg e Tarjan [GT90] tem complexidade $O(nm \log \frac{n^2}{m})$ por fase escalonada e complexidade total $O(nm \log \frac{n^2}{m} \log(nC))$.

No contexto do PD, os vértices ativos são os vértices não emparelhados ou em múltiplos emparelhamentos. As operações `enviar/renomear` são interpretadas neste contexto como:

`enviar` consiste em emparelhar um vértice ativo com exatamente um único vértice. Se o vértice ativo em questão pertence à partição X, esta operação resulta em emparelhá-lo a um vértice de Y através de uma de suas arestas admissíveis. O vértice de Y pode ser um vértice emparelhado. Se o vértice em questão pertence à partição Y, esta operação consiste em eliminar do emparelhamento uma das suas arestas admissíveis.

`renomear` aplica-se a um vértice ativo que não possui arestas admissíveis. Esta operação consiste em definir o potencial do vértice ativo v como

$$\pi(v) = \begin{cases} \max\limits_{(v,y)\in E(\mathcal{M})} \{\pi(y) - c_{vy}\} & \text{caso o vértice } v \in X \\ \max\limits_{(x,v)\in E(\mathcal{M})} \{\pi(x) - c_{xv} - \epsilon\} & \text{caso contrário} \end{cases}$$

Após a aplicação desta operação no mínimo uma aresta admissível foi gerada.

Em uma única iteração escalonada um vértice não pode ser renomeado mais do que $O(n)$ vezes. Desta maneira, o número de operações de envio em uma única fase é limitado por $O(nm)$. A complexidade de cada iteração escalonada do algoritmo é $O(nm)$. Desta maneira, a complexidade do algoritmo de escalonamento de custos para o PD é $O(nm \log(nC))$.

Gabow e Tarjan [GT89a] propuseram um algoritmo baseado em otimalidade aproximada utilizando outras técnicas com complexidade $O(\sqrt{n}m \log(nC))$. Para as instâncias satisfazendo a suposição de similaridade este algoritmo é o melhor algoritmo de tempo polinomial fraco existente para o PD. Posteriormente, Goldberg e outros [GPV93] desenvolveram um algoritmo com a mesma complexidade baseado nas operações `enviar/renomear`.

A complexidade dos algoritmos descritos neste capítulo para o PD e para o PFCM são exibidos na tabela 4.1.

Complexidade dos algoritmos		
algoritmo	PFCM	PD
1	$O(\min\{n^2m^2\log(nC), n^2m^3\log n\})$	$O(\min\{n^2m^2\log(nC), n^2m^3\log n\})$
2	$O(\min\{n^2m\log(nC), n^2m^2\log n\})$	$O(nm\log(nC))$
3	$O(\min\{n^3D, D\sqrt{n^3}m\log(nC)\})$	$O(\min\{n^3, \sqrt{n^3}m\log(nC)\})$
4	$O(\min\{nU, nC\}\{n^3\})$	$O(nm + n^2\log n)$
5	$O(nm\log\frac{n^2}{m}\log(nC))$	$O(\sqrt{n}m\log(nC))$
1 - algoritmo de cancelamento de ciclos		
2 - algoritmo simplex de rede		
3 - algoritmo de caminhos mais curtos sucessivos		
4 - algoritmo primal-dual		
5 - algoritmo de escalonamento de custos		

Tabela 4.1: *Algoritmos descritos para o* PFCM *e* PD

Capítulo 5

Problemas paramétricos

Neste capítulo, estudaremos as variantes paramétricas do PD e do PFCM. Até agora estudamos esses problemas supondo que os custos das arestas são constantes. Na variante paramétrica que iremos estudar, o custo de uma aresta e é dado por uma função linear do tipo $c_e = c_e^\circ + \lambda c_e^\bullet$, onde c_e° e $c_e^\bullet$ são constantes e λ é o *parâmetro*.

Dado um intervalo de variação de λ, que iremos denotar por $[\underline{\lambda}, \overline{\lambda}]$, nosso objetivo será encontrar pelo menos uma solução dentre as soluções ótimas possíveis para cada valor do parâmetro dentro desse intervalo.

Se fixamos um valor para λ obtemos uma instância do problema não-paramétrico relacionado. Para cada instância do problema não-paramétrico podem existir diversas soluções ótimas estruturalmente distintas, todas com o mesmo custo. O problema paramétrico pode ser visualizado como sendo constituído de uma série de instâncias do problema não-paramétrico. Uma forma trivial de se resolver um problema paramétrico consiste em se aplicar um algoritmo para o problema não-paramétrico a cada instância definida fixando o valor de λ. É possível que instâncias relacionadas a valores distintos de λ apresentem soluções ótimas com a mesma estrutura. O custo destas soluções entretanto deve diferir, pois o custo das arestas dependem do valor do parâmetro.

Pode-se imaginar que podemos conceber algoritmos eficientes para o problema paramétrico que compute apenas as soluções ótimas distintas ao invés de simplesmente resolver uma série de instâncias do problema não-paramétrico. O tema deste capítulo é a concepção e o estudo de algoritmos para resolver o PD paramétrico da melhor maneira possível.

5.1 Conceitos básicos

Seja $\mathcal{P}$ um problema combinatório onde os dados do problema são constantes $\alpha_1, \ldots, \alpha_\beta$. Denominamos por $\mathcal{S}$ uma solução factível para o problema e definimos o valor desta

solução por

$$V_{\mathcal{S}} = F(\alpha_1, \ldots, \alpha_\beta)$$

Considere que sabemos como calcular a solução ótima $\mathcal{S}^*$ de menor valor. A solução ótima para o problema $\mathcal{P}$ é expressa por

$$V_{\mathcal{S}^*} = \min_{\forall \mathcal{S}} V_{\mathcal{S}}$$

Suponha que generalizamos o problema $\mathcal{P}$ parametrizando os seus dados. O problema paramétrico denotado por $\mathcal{P}'$ possui dados $\alpha_1 + \lambda\gamma_1, \ldots, \alpha_\beta + \lambda\gamma_\beta$ em função de um parâmetro λ. O valor de uma solução factível $\mathcal{S}$ para $\mathcal{P}'$ varia linearmente com o parâmetro

$$V_{\mathcal{S}}(\lambda) = F(\alpha_1 + \lambda\gamma_1, \ldots, \alpha + \lambda\gamma_\beta)$$

Deseja-se agora resolver o problema para todos os valores de λ no intervalo $[\underline{\lambda}, \overline{\lambda}]$. A solução ótima $\mathcal{S}^*$ consiste em obter um conjunto de soluções factíveis tal que para qualquer valor fixado para λ exista pelo menos uma solução de valor mínimo pertencente ao conjunto. O valor da solução ótima é definido pela expressão

$$V^*_{\mathcal{S}}(\lambda) = \min_{\forall \mathcal{S} \in \mathcal{S}^*} V_{\mathcal{S}}(\lambda)$$

Os problemas paramétricos podem ser generalizados para o caso em que os dados do problema estão em função linear de vários parâmetros. Neste texto, nos restringiremos à estudar o caso do PD paramétrico em função de um único parâmetro.

Gusfield [Gus83] demonstra que a função $V_{\mathcal{S}^*}(\lambda)$ é bem-definida, linear por partes e convexa em função de λ. A função é composta por segmentos lineares que representam a otimalidade de cada uma das soluções do conjunto $\mathcal{S}^*$. A convexidade da função decorre do fato de que o segmento associado a uma solução do conjunto $\mathcal{S}^*$, para uma determinada faixa de valores do parâmetro deve ser dominado (ser o de menor valor) pelos segmentos das demais soluções factíveis.

Os valores do parâmetro para o qual a inclinação de $V_{\mathcal{S}^*}(\lambda)$ varia são denominados **pontos-de-quebra**. Os pontos-de-quebra são um importante conceito na resolução dos problemas paramétricos. O intervalo da otimalidade de uma solução em $\mathcal{S}^*$ é limitado por pontos-de-quebra. Alguma solução ótima entre dois pontos-de-quebra consecutivos é ótima para toda a faixa de valores do parâmetro entre estes pontos-de-quebra.

Cada problema paramétrico resulta na descrição da função $V^*_{\mathcal{S}}(\lambda)$ associada ao problema. Esta função pode ser completamente descrita determinando os pontos-de-quebra $\omega = \{\omega_1, \ldots, \omega_\eta\}$ da função, mais um conjunto de soluções $\mathcal{S}^* = \{\mathcal{S}_0, \ldots, \mathcal{S}_\eta\}$ tal que $\mathcal{S}_k$ é ótima no intervalo $[\omega_{k-1}, \omega_k]$. Conhecendo-se os pontos-de-quebra, pode-se estabelecer as soluções resolvendo instâncias do problema não-paramétrico dentro dos intervalos. Se ω e $\mathcal{S}^*$ são conhecidos, a solução ótima para um determinado valor do parâmetro pode ser

determinada eficientemente ao invés de ser computada. Necessita-se apenas encontrar a região que contém o valor especificado e recuperar a estrutura da solução ótima para esta região.

Na resolução dos problemas paramétricos busca-se obter cada solução presente no conjunto de soluções em tempo polinomial por ponto-de-quebra. Busca-se, se possível, explorar a relação entre a seqüência de instâncias processadas e a estrutura do problema avaliado, para obter cada solução num tempo assintótico inferior à complexidade de resolução de uma instância do problema não-paramétrico relacionado.

5.2 Abordagens na literatura

Algumas técnicas propostas na literatura para a descrição da função $V_{\mathcal{S}^*}(\lambda)$ se aplicam à resolução de diversos problemas combinatórios paramétricos.

Eisner e Severance [ES76] propuseram um método genérico, para obter todos os pontos-de-quebra e descrever uma função $V_{\mathcal{S}^*}(\lambda)$ linear 'por partes' num certo intervalo $[\underline{\lambda}, \overline{\lambda}]$. O método, o qual denominamos *ES*, é simples e pode ser aplicado para obter cada solução do conjunto de soluções ótimas em tempo polinomial por ponto-de-quebra. Pelo método *ES*, a complexidade para se determinar cada ponto-de-quebra é igual à complexidade para se resolver uma instância do problema não-paramétrico.

O método *ES* consiste em encontrar a solução ótima para os parâmetros nos extremos do intervalo $[\underline{\lambda}, \overline{\lambda}]$, respectivamente $\mathcal{S}'$ e $\mathcal{S}''$. Se o valor da solução $\mathcal{S}'$ em $\overline{\lambda}$ é igual ao valor de $\mathcal{S}''$ neste ponto $V_{\mathcal{S}'}(\overline{\lambda}) = V_{\mathcal{S}''}(\overline{\lambda})$, ou $V_{\mathcal{S}'}(\underline{\lambda}) = V_{\mathcal{S}''}(\underline{\lambda})$, temos que para o intervalo em questão existe uma única solução ótima. A função $V_{\mathcal{S}^*}(\lambda)$ é descrita por um único segmento $\mathcal{S}^* = \{\mathcal{S}'\}$ (ou $\{\mathcal{S}''\}$) e $\omega = \emptyset$.

Se a solução $\mathcal{S}'$ não é ótima em $\overline{\lambda}$, existe um ponto específico único μ no intervalo $[\underline{\lambda}, \overline{\lambda}]$ em que o valor de $\mathcal{S}'$ e $\mathcal{S}''$ são iguais. Para o intervalo $[\underline{\lambda}, \mu]$ a solução $\mathcal{S}'$ possui custo inferior a $\mathcal{S}''$. No intervalo $[\mu, \overline{\lambda}]$ a situação inversa ocorre. Se as soluções $\mathcal{S}'$ e $\mathcal{S}''$ são ótimas em μ então μ é um ponto-de-quebra de $V_{\mathcal{S}^*}(\lambda)$. A função $V_{\mathcal{S}^*}(\lambda)$ neste caso é descrita pelo conjunto $\mathcal{S}^* = \{\mathcal{S}', \mathcal{S}''\}$ e pelo ponto-de-quebra $\omega = \{\mu\}$.

Se a solução ótima $\mathcal{S}^\circ$ em μ não for $\mathcal{S}'$ ou $\mathcal{S}''$, $\mathcal{S}^\circ$ define um segmento da função $V_{\mathcal{S}^*}(\lambda)$ entre os segmentos de $\mathcal{S}'$ e $\mathcal{S}''$. O método ES prossegue determinando os pontos-de-quebra de $V_{\mathcal{S}^*}(\lambda)$ nos subintervalos $[\underline{\lambda}, \mu]$ e $[\mu, \overline{\lambda}]$. A solução ótima nos extremos dos subintervalos são conhecidas, de maneira que a avaliação descrita para obter os pontos-de-quebra em $[\underline{\lambda}, \overline{\lambda}]$ pode ser aplicada recursivamente à cada um dos subintervalos.

A iteração básica do método *ES* consiste na resolução de uma instância do problema não-paramétrico em pontos específicos de λ. Estes pontos podem ser calculados através da função linear que define o valor de cada solução. O número de instâncias do problema não-paramétrico resolvidas aplicando a solução de Eisner e Severance não é superior ao

dobro do número de pontos-de-quebra de $V_{\mathcal{S}^*}(\lambda)$. O método *ES* obtém ω e $\mathcal{S}^*$ em tempo polinomial por ponto-de-quebra se o problema não-paramétrico possui resolução em tempo polinomial.

Posteriormente, Gusfield [Gus83] propôs outro método genérico para os problemas combinatórios paramétricos. O método de Gusfield, o qual denominamos *DG* , é aplicável à várias classes de problemas e mais adequado para se estender aos problemas com informações em função de vários parâmetros. O método *DG* possui as mesmas características desejáveis do método *ES* e alguns benefícios adicionais.

Uma vantagem da aplicação do método *DG* é que este descreve a função $V_{\mathcal{S}^*}(\lambda)$ obtendo os pontos-de-quebra de ω consecutivamente. Esta característica possibilita que este algoritmo seja aplicado de forma on-line. A primeira solução do conjunto $\mathcal{S}^*$ é definida pela solução ótima S' no extremo inferior do intervalo. O primeiro ponto-de-quebra estritamente maior que $\underline{\lambda}$ é determinado. Em seguida os pontos-de-quebra subsequentes são determinados até que todos os pontos dentro do intervalo sejam estabelecidos.

Seja A um algoritmo para o problema $\mathcal{P}$ e a sua entrada um conjunto de variáveis simbólicas. Considere sem perda de generalidade que A equivale à uma árvore binária de decisão B. Cada ramificação de B é uma comparação entre variáveis simbólicas. Assumimos que esta comparação é a relação $<$ e $\geq$ entre as variáveis. Seja ω_1 o primeiro ponto-de-quebra de $V_{\mathcal{S}^*}(\lambda)$. A aplicação do algoritmo para o parâmetro $\lambda = \omega_1$ define um único caminho em B. A idéia do algoritmo de Gusfield é simular a execução do algoritmo A e identificar o caminho associado à instância do problema definido por $\lambda = \omega_1$ sem conhecer o valor ω_1.

Suporemos que em algum vértice de B os valores das variáveis computadas através do caminho até este vértice são expressões lineares bem-definidas em λ. Um requisito simples para satisfazer tal característica é restringir o algoritmo A a realizar operações aritméticas somente de adição e subtração. Multiplicações são permitidas desde que envolvam variáveis e constantes. Esta restrição não é forte sendo que a maioria dos algoritmos combinatórios possuem esta característica.

Na simulação de A considere a comparação realizada em alguma ramificação de B entre as variáveis simbólicas g e q. Indicaremos o valor de g e q computadas até esta ramificação da árvore por $g(\lambda)$ e $q(\lambda)$. O resultado da comparação definirá a próxima aresta do caminho associado a ω_1. O objetivo é descobrir qual das relações $g(\omega_1) < q(\omega_1)$ ou $g(\omega_1) \geq q(\omega_1)$ é verdadeira. Seja λ° o único valor no qual $g(\lambda^\circ) = q(\lambda^\circ)$. Seja $\mathcal{S}''$ a solução ótima em λ°. Suponha que $g(\lambda) < q(\lambda)$ para os valores do parâmetro a esquerda de λ°. Se $V_{\mathcal{S}''}(\lambda^\circ) < V_{\mathcal{S}'}(\lambda^\circ)$ então $\mathcal{S}'$ não e ótima em λ°. Desta forma, o ponto-de-quebra procurado satisfaz $w_1 < \lambda^\circ$ e a relação $g(\omega^1) < q(\omega^1)$ se aplica. Caso contrário, $g(\omega^1) \geq q(\omega^1)$. Conclusões similares podem ser obtidas no caso em que $g(\lambda) \geq q(\lambda)$ para os valores do parâmetro a esquerda de λ°. Provas da corretude deste algoritmo são

demonstradas no texto de Gusfield.

Assim como o método ES, o método proposto por Gusfield pode determinar cada ponto-de-quebra em tempo polinomial se existem algoritmos de tempo polinomial para o problema não-paramétrico. Cada ramificação da árvore de decisão corresponde à resolução de uma instância do problema. Se o número de comparações realizadas pelo algoritmo é limitado polinomialmente, cada ponto-de-quebra pode ser obtido em tempo polinomial. Em seu trabalho Gusfield discute como este método pode ser generalizado eficientemente para funções com mais de um parâmetro.

Se $\mathcal{P}$ é um problema de programação linear então o método simplex paramétrico pode ser aplicado. Este método é aplicado constantemente em Pesquisa Operacional. Este método possui deficiências para a aplicação aos problemas de Fluxo em Redes. O método não possui complexidade polinomial e sofre do problema de degeneração presente no algoritmo simplex.

5.3 O problema do fluxo de custo mínimo paramétrico

Existem várias abordagens na literatura para os problemas de Fluxos em Redes paramétricos. Dentre estas, uma em especial pode ser aplicada diretamente ao PD paramétrico. Esta abordagem é a solução proposta por Srinivasan e Thompson [ST72] para o PFCM paramétrico. O **problema do fluxo de custo mínimo paramétrico** é a generalização do clássico PFCM com os dados do problema, como os custos, definidos em função de um parâmetro λ.

Srinivasan e Thompson propuseram soluções para vários casos do PFCM paramétrico com base no algoritmo simplex de rede descrito na seção 4.1.2. O algoritmo de Srinivasan e Thompson pode ser aplicado para resolver o PFCM paramétrico com custos, capacidade ou desbalanceamento dos vértices parametrizados. A solução para o PFCM paramétrico com os custos das arestas parametrizados pode ser naturalmente aplicado ao PD paramétrico.

No PFCM paramétrico com parametrização dos custos, definimos os custos das arestas em função do parâmetro λ variando no intervalo $[\underline{\lambda}, \overline{\lambda}]$ como $c_e = c_e^\circ + \lambda c_e^\bullet$, onde c° e $c^\bullet$ são constantes inteiras. Denotando por G_λ o grafo gerado fixando o parâmetro λ. O custo do fluxo ótimo em G_λ é definido pela função

$$c_{\mathcal{F}^*}(\lambda) = \min_{\forall \mathcal{F}} \sum_{e \in \mathcal{F}} (c_e^\circ + \lambda c_e^\bullet) f_e$$

O algoritmo proposto por Srinivasan e Thompson, o qual denominamos *ST*, se baseia na condição de otimalidade de custo reduzido especializada para o algoritmo simplex de rede.

Seja $\mathcal{A}' = (\mathcal{T}', \mathcal{L}', \mathcal{U}')$ a estrutura de árvore espalhada para o fluxo de custo mínimo em $G_{\underline{\lambda}}$. Denotamos por π° e $\pi^\bullet$ os potenciais dos vértices que satisfazem a condição de otimalidade para a árvore espalhada $\mathcal{T}'$ com arestas de custo respectivamente, c° e $c^\bullet$.

Srinivasan e Thompson demonstram um importante resultado referente aos potenciais de $\mathcal{A}'$. O resultado descrito a seguir não será formalmente provado apenas daremos uma breve explicação.

Lema 5.3.1 *Se π° denota os potenciais para o qual as arestas da árvore espalhada $\mathcal{T}'$ possuem custo reduzido zero quando c°_e são os custos das arestas, e $\pi^\bullet$ os potenciais quando $c^\bullet_e$ são os custos das arestas, então para os potenciais $\pi^\circ + \lambda\pi^\bullet$ as arestas de $\mathcal{T}'$ possuem custo reduzido zero quando $c^\circ_e + \lambda c^\bullet_e$ são os custos das arestas.*

Explicação:Pela condição de otimalidade de custo reduzido temos que as expressões

$$c^{\pi^\circ}_{vw} = c^\circ_{vw} - \pi^\circ(v) + \pi^\circ(w) = 0$$
$$c^{\pi^\bullet}_{vw} = c^\bullet_{vw} - \pi^\bullet(v) + \pi^\bullet(w) = 0$$

se verificam para todas as arestas de $\mathcal{T}'$. Assim, $c^{\pi^\circ}_{vw} + \lambda c^{\pi^\bullet}_{vw} = 0 \ \ \forall (v,w) \in \mathcal{T}'$. De outra maneira podemos declarar a expressão anterior como:

$$(c^\circ_{vw} - \pi^\circ(v) + \pi^\circ(w)) \ + \ \lambda \ (c^\bullet_{vw} - \pi^\bullet(v) + \pi^\bullet(w)) = 0$$

Reorganizando estas informações, podemos verificar que:

$$(c^\circ_{vw} + \lambda c^\bullet_{vw}) \ - \ (\pi^\circ(v) + \lambda\pi^\bullet(v)) \ + \ (\pi^\circ(w) + \lambda\pi^\bullet(w)) = 0$$
$$c'_{vw} \ - \ \pi'(v) \ + \ \pi'(w) = 0 \qquad\qquad \forall (v,w) \in \mathcal{T}'$$

O resultado de Srinivasan e Thompson provê subsídios para caracterizar a otimalidade da estrutura de árvore espalhada quando variamos o parâmetro. Se definimos os custos das arestas como $c^* = c^\circ + \lambda^* c^\bullet$, para os potenciais $\pi^* = \pi^\circ + \lambda^*\pi^\bullet$ as arestas da árvore espalhada satisfazem à condição de otimalidade. As arestas de $\mathcal{T}'$ possuem custo reduzido zero com respeito à π^* para qualquer valor do parâmetro. A otimalidade da estrutura $\mathcal{A}'$ continuará para valores do parâmetro subsequentes enquanto nenhuma aresta violar a condição de otimalidade para os potenciais π^*. Se alguma aresta viola a otimalidade, a mesma deve ser uma aresta restrita. Pode-se assim estabelecer a otimalidade de uma estrutura de árvore espalhada a partir da avaliação da otimalidade das arestas restritas.

Iremos denotar por λ_{vw} o maior valor do parâmetro para o qual a aresta (v,w) satisfaz

a condição de otimalidade. Formalmente definimos λ_{vw} como:

$$\lambda_{vw} = \begin{cases} \forall (v,w) \in \mathcal{L}' & \begin{cases} \frac{-c^{\pi^\circ}_{vw}}{c^{\pi^\bullet}_{vw}} & \text{se } c^{\pi^\bullet}_{vw} < 0 \\ \infty & \text{caso contrário} \end{cases} \\ \forall (v,w) \in \mathcal{U}' & \begin{cases} \frac{-c^{\pi^\circ}_{vw}}{c^{\pi^\bullet}_{vw}} & \text{se } c^{\pi^\bullet}_{vw} > 0 \\ \infty & \text{caso contrário} \end{cases} \end{cases}$$

Seja λ° o maior valor do parâmetro para o qual a estrutura de árvore espalhada $\mathcal{A}'$ satisfaz a condição de otimalidade. O valor de λ° pode ser determinado a partir dos valores de λ_{vw}:

$$\lambda^\circ = \min\{\lambda_{vw} \quad (v,w) \in \mathcal{L}' \bigcup \mathcal{U}'\}$$

Seja (i,j) a aresta que define o valor de $\lambda^\circ = \lambda_{ij}$. Pode-se observar que o custo reduzido da aresta (i,j) para $\lambda = \lambda^\circ$ é $c^{\pi^\circ}_{ij} + \lambda^\circ c^{\pi^\circ}_{ij} = 0$. Para o intervalo abrangendo $[\underline{\lambda}, \lambda^\circ]$, a estrutura de árvore espalhada $\mathcal{A}'$ permanece sendo ótima. A partir de λ° a aresta (i,j) violará a condição de otimalidade. Uma outra estrutura de árvore espalhada satisfazendo a condição de otimalidade existe para os valores do parâmetro superiores à λ°.

Não necessariamente λ° define um ponto-de-quebra da função $c_{\mathcal{F}^*}(\lambda)$. Isto ocorre porque λ° não caracteriza o maior valor do parâmetro para o qual a solução representada pela estrutura $\mathcal{A}'$ permanece ótima. Este valor apenas indica que a estrutura $\mathcal{A}'$ não será ótima a partir de λ°. Como discutido na seção 4.1.2, uma solução para o PFCM pode ser representada por várias estruturas de árvore espalhada. Se λ° não define um ponto-de-quebra então existe uma outra estrutura de árvore espalhada para representar a mesma solução a partir de λ°.

Suponha que um pivotamento é realizado sobre $\mathcal{A}'$ com a aresta (i,j) como sendo a aresta de entrada. A inserção da aresta em $\mathcal{A}'$ provoca o surgimento de um ciclo de custo zero. Se um ciclo é gerado, o aumento do fluxo através do mesmo servirá para obtermos uma nova estrutura de árvore espalhada $\mathcal{A}^\circ$.

A estrutura $\mathcal{A}^\circ$ assim como $\mathcal{A}'$ satisfaz à condição de otimalidade para os potenciais $\pi^\circ + \lambda\pi^\circ$ para λ°. Quando a solução representada por $\mathcal{A}'$ for não degenerada, o pivotamento criará um ciclo orientado de custo zero. O aumento do fluxo através do ciclo orientado resultará num novo fluxo. Neste caso se $\lambda^\circ \neq \underline{\lambda}$ temos que um ponto-de-quebra foi encontrado.

Procedendo como antes, podemos determinar o intervalo para o qual a nova estrutura $\mathcal{A}^\circ$ permanecerá ótima. Por este método, pode-se continuamente obter todos os intervalos de cada uma das soluções distintas do problema dentro do intervalo estipulado $[\underline{\lambda}, \overline{\lambda}]$.

5.3.1 O comportamento da abordagem *ST*

A complexidade do algoritmo *ST*, assim como a do algoritmo simplex de rede, depende fundamentalmente do número de pivotamentos realizados. As soluções degeneradas ocasionam iterações desnecessárias. Nestas iterações nenhum ponto-de-quebra é encontrado. Apesar do pivotamento poder ser realizado eficientemente, se o algoritmo necessitar de várias iterações para determinar um ponto-de-quebra esta abordagem torna-se inviável.

Não foi possível encontrar nos textos consultados para o estudo desta abordagem informações que relatem problemas ou dificuldades para garantir a execução finita do algoritmo para esta abordagem[1]. A abordagem de Srinivasan e Thompson enfrenta o problema de degeneresçência. Não se verificou na literatura nenhuma regra que deva ser aplicada na escolha das arestas utilizadas na realização dos pivotamentos. Nenhuma análise sobre a complexidade computacional do algoritmo *ST* foi verificada na literatura. Não se pode afirmar com segurança que o número de iterações para obter cada ponto-de-quebra pode ser limitado polinomialmente.

Alguns fatores nos levam a acreditar que o algoritmo *ST* é ineficiente na resolução do PD paramétrico. O motivo para este entendimento está na estrutura muito particular do problema. A árvore espalhada associada a qualquer emparelhamento é altamente degenerada. Este grau elevado de degenerescência implicará num elevado número de iterações desnecessárias. Para perceber como este grau de degenerescência afeta a execução do algoritmo, observe que todas as arestas na árvore são restritas (nunca existem duas arestas entre dois vértices na árvore). A maioria dos pivotamentos realizados não produz um ciclo orientado. Sem a existência de um ciclo orientado, nenhuma quantidade de fluxo pode ser enviada através do ciclo. Como conseqüência, a nova estrutura obtida representa o mesmo emparelhamento anterior. Este tipo de iteração é desnecessária não trazendo novas informações sobre os pontos-de-quebra.

A complexidade para a obtenção de cada solução do PD paramétrico não é superior à complexidade para uma solução do PFCM paramétrico. Porém, se imagina que o comportamento empírico da aplicação do algoritmo *ST* a uma instância do PFCM deve apresentar melhores resultados. Algumas das arestas na árvore espalhada na instância do PFCM serão irrestritas. Desta forma, a aresta e a sua reversa estarão presentes na árvore espalhada. Esta característica aumenta a probabilidade de se gerar um ciclo orientado com a inserção de uma nova aresta.

[1] Infelizmente não se conseguiu ter acesso ao artigo de Srinivasan e Thompson. As informações obtidas sobre esta abordagem vieram do texto de Ahuja e outros [AMO93]

5.4 Abordagens para problemas de Fluxos em Redes específicos

As abordagens gerais para os problemas paramétricos apresentadas na seção 5.2 não exploram as peculiaridades de cada problema. As abordagens de Eisner e Severance, e posteriormente Gusfield, são simplesmente métodos de busca aplicados à determinação do valor de cada ponto-de-quebra. Ambos os métodos estão baseados na resolução de instâncias do problema não-paramétrico relacionado. Soluções mais eficientes para alguns problemas de Fluxos em Redes foram obtidas quando se estudou intensamente a estrutura do problema paramétrico em questão.

Esta seção apresenta resultados para outros problemas, fruto de estudos cujo objetivo era obter soluções que pudessem ser aplicadas ao PD paramétrico que fossem melhores do que o algoritmo *ST*.

Foram estudadas a solução de Gallo e outros [GGT89] para o problema do fluxo máximo paramétrico e as soluções de Karp e Orlin [KO81] e Young e outros [YTO91] para o problema do caminho mais curto paramétrico. A abordagem de Gallo e outros se mostrou pouco favorável para aplicação ao PD paramétrico, de maneira que esta não será descrita. A abordagem para o problema do caminho mais curto paramétrico se mostrou útil. Por este motivo, descrevemos esta abordagem na próxima seção.

5.4.1 O problema do caminho mais curto paramétrico

Karp e Orlin [KO81] propuseram algoritmos eficientes para uma classe especial do problema do caminho mais curto paramétrico. Nesta classe especial, a variação dos custos das arestas em relação ao parâmetro λ está restrita aos valores 0 e -1. Problemas paramétricos com esta restrição, **problemas paramétricos 0-1**, possuem bastante interesse na literatura sendo freqüentemente abordados.

Seja $G = (V, E)$ um grafo orientado ponderado incluindo um vértice especial s. Os custos das arestas são definidos em função do parâmetro λ como $c_e = c_e^\circ + \lambda c_e^\bullet$, onde c_e° é uma constante inteira e $c_e^\bullet \in \{-1, 0\}$. Um caminho entre o vértice s e o vértice $v \in V$ é denotado por P_v. Supõe-se que todos os vértices de G são alcançáveis de s, ou seja, P_v é bem-definido. O **problema do caminho mais curto paramétrico** consiste em determinar a árvore de caminhos mais curtos $\mathcal{T}$ em G_λ para cada valor de λ em $[\underline{\lambda}, \overline{\lambda}]$ tal que os caminhos mais curtos são bem-definidos.

O custo da árvore de caminhos mais curtos em função do parâmetro é definido como

$$c_{\mathcal{T}^\bullet}(\lambda) = \min_{\forall \mathcal{T}} \sum_{e \in \mathcal{T}} c_e^\circ + \lambda c_e^\bullet$$

onde $\lambda \in [\underline{\lambda}, \overline{\lambda}]$ ou até o valor máximo do parâmetro para o qual os caminhos mais curtos são bem-definidos. O caminho na árvore $\mathcal{T}$ entre s e $v \in V$ é denotado por t_v.

Karp e Orlin propuseram duas soluções distintas para resolver o PCMC paramétrico. A mais importante para este estudo é a solução baseada em árvores balanceadas.

Seja $\mathcal{T}'$ a árvore de caminhos mais curtos em $G_{\underline{\lambda}}$. A árvore $\mathcal{T}'$ permanecerá ótima para valores de λ subsequentes enquanto os caminhos t'_v em $\mathcal{T}'$ (de s a v) permanecem os menores entre estes vértices.

Suponha que em algum momento a árvore $\mathcal{T}'$ perde a otimalidade para a árvore $\mathcal{T}''$. No mínimo um caminho t''_v tornou-se menor que o correspondente caminho t'_v em $\mathcal{T}'$, para algum $v \in V$. O tamanho de t''_v decresceu mais rapidamente que o tamanho de t'_v. Assim o caminho t''_v deve ter necessariamente mais arestas parametrizadas que t'_v.

Para um valor único $\mu > \underline{\lambda}$, o tamanho de t''_v foi igual ao tamanho de t'_v. Em G_μ tanto $\mathcal{T}'$ quanto $\mathcal{T}''$ são árvores de caminhos mais curtos. A partir de μ apenas $\mathcal{T}''$ continuará mantendo a otimalidade. O valor μ delimita a otimalidade entre duas árvores de caminhos mais curtos sendo portanto um ponto-de-quebra.

O algoritmo proposto por Karp e Orlin consiste em determinar o primeiro caminho P_v não presente em $\mathcal{T}'$ que irá se igualar em custo ao seu correspondente caminho t'_v na árvore e para qual valor do parâmetro μ isto ocorre. O valor μ caracteriza o término da otimalidade de uma árvore de caminhos mais curtos e o início da otimalidade de outra. A nova árvore de caminhos mais curtos a partir de μ pode ser obtida excluindo todas as arestas da árvore que incidem sobre os vértices de P_v e adicionando o caminho P_v a $\mathcal{T}'$.

O algoritmo termina quando a árvore permanecer ótima para todos os valores do parâmetro subsequentes ou até que os caminhos mais curtos sejam bem-definidos. Para reduzir o número de caminhos analisados, o algoritmo avalia apenas os caminhos formados por um subcaminho t_u na árvore de s a u, seguido por uma aresta $e = (u, v)$ não presente na árvore. Para cada aresta e, o algoritmo determina em que momento α_e este caminho se igualará ao correspondente caminho na árvore. O valor α_e pode ser calculado em tempo constante se o custo do menor caminho de s a v, $\forall v \in V$, e o número de arestas parametrizadas no caminho são mantidas. O objetivo deste texto não é discutir todos os detalhes do algoritmo, mas apenas apresentar as idéias no qual se baseia a abordagem de Karp e Orlin. Informações detalhadas sobre o algoritmo podem ser obtidas nos textos de Karp e Orlin e no texto de Young e outros.

Karp e Orlin demonstraram que o número máximo de soluções distintas para esta classe particular do PCMC paramétrico é $O(n^2)$ árvores. O algoritmo *KO* pode ser estendido para resolver o PCMC paramétrico com constantes $c^\bullet$ inteiras. Nesta situação, entretanto, o número de árvores encontradas não é limitado polinomialmente. A complexidade original do algoritmo *KO* era $O(nm \log n)$, se as informações sobre α_e fossem manipuladas em árvores balanceadas. Posteriormente, Young e outros melhoraram a complexidade do

algoritmo KO para $O(nm + n^2 \log n)$. Alterando o algoritmo para utilizar a estrutura de dados de lista de prioridade de Fibonacci [FT87].

Um importante resultado do trabalho de Karp e Orlin é a redução do problema do ciclo de média mínima-(PCMM) ao PCMC paramétrico. O PCMM, definido na seção 4.1.1, pode ser reduzido a um caso particular do PCMC paramétrico onde os custos das arestas decrescem uma unidade com o aumento do parâmetro λ. O relacionamento verificado entre estes problemas tem aplicações práticas. O algoritmo proposto por Karp e Orlin é considerado o melhor algoritmo empírico para determinar um ciclo de média mínima em um grafo. Apesar do resultado de Karp e Orlin não implicar em melhoramentos do ponto de vista teórico para o PCMM, ele mostra uma relação que também foi verificada neste estudo entre os problemas envolvendo ciclos e os problemas paramétricos.

Redução do PCMM ao PCMC paramétrico [2] Dada uma instância do problema do ciclo de média mínima $G = (V, E)$, construa uma instância $G' = (V', E')$ do problema dos caminhos mais curtos paramétricos. Defina o custo das arestas $e' \in E'$ do grafo como $c_{e'} = c_e - \lambda$, onde c_e é o custo da aresta relacionada a e' em G. Pode-se obter a solução ótima para o PCMM a partir da solução ótima para o PCMC paramétrico no intervalo $[-\infty, \infty]$.

Seja o conjunto de árvores $\mathcal{T}_0, \ldots, \mathcal{T}_\eta$ e a sequência de pontos-de-quebra $\lambda_0, \ldots, \lambda_\eta$ a solução do PCMC paramétrico. Cada árvore do conjunto é a árvore de caminhos mais curtos para uma faixa de valores do parâmetro compreendida entre dois pontos-de-quebra.

Para os valores do parâmetro superiores a λ_η os caminhos mais curtos não são bem-definidos devido à presença de um ciclo negativo, denotaremos este ciclo por $\mathcal{C}$. Em G_{λ_η} entretanto não existem ciclos de custo negativo. No mínimo um ciclo de custo zero, o ciclo $\mathcal{C}$, está presente. Da condição de otimalidade de Custo Reduzido (seção 3.5.2), sabemos que existe um conjunto de potenciais π para o qual todas as arestas possuem custo reduzido não-negativo se os caminhos mais curtos são bem-definidos. Os potenciais não afetam os custos dos ciclos de maneira que $\mathcal{C}$ é um ciclo de custo zero em relação aos custos reduzidos. A média do ciclo $\mathcal{C}$ em G_{λ_η} é zero. Nenhum ciclo em $G_{\lambda_{\eta+1}}$ pode ter média negativa resultando que $\mathcal{C}$ é um ciclo de média mínima para esta instância.

Cada aresta em G_{λ_η} teve seu custo reduzido em λ_η em relação a G. Em relação aos potenciais π todas as arestas de G possuem custo reduzido positivo. O menor custo reduzido de uma aresta de G em relação à π é λ_η. Todas as arestas de $\mathcal{C}$ tem custo reduzido igual a λ_η em G, o mesmo valor da média deste ciclo. Novamente nenhum ciclo pode ter média inferior à média de $\mathcal{C}$. Desta forma, $\mathcal{C}$ é o ciclo de média mínima procurado.

[2] A redução do PCMM ao PCMC paramétrico declarada a seguir é baseada no texto de Young e outros [YTO91]

5.5 Uma nova abordagem para o problema da designação paramétrico

As idéias presentes na abordagem do PCMC paramétrico podem ser aplicadas numa nova abordagem para o PD paramétrico. No contexto do PD, a perda da otimalidade de um emparelhamento é caracterizada pelo surgimento de um ciclo negativo no grafo residual. A resolução do PD paramétrico envolve estabelecer a faixa de valores do parâmetro em que o grafo residual, associado à um emparelhamento ótimo, permanece sem ciclos de custo negativo.

Os teoremas a seguir provêem fundamentos para a construção de algoritmos para o PD paramétrico.

Seja $\mathcal{M}$ um emparelhamento ótimo para o valor λ' do parâmetro.

Lema 5.5.1 *Se para um valor $\lambda'' > \lambda'$ os caminhos mais curtos são bem-definidos no grafo residual de $\mathcal{M}$ então o emparelhamento $\mathcal{M}$ é ótimo no intervalo $[\lambda', \lambda'']$.*

Prova: Se o emparelhamento $\mathcal{M}$ é ótimo em λ', temos que $G(\mathcal{M})$ não possui ciclos de custo negativo em λ'. Pelo mesmo motivo, o emparelhamento $\mathcal{M}$ é ótimo em λ''. O custo dos ciclos em $G(\mathcal{M})$ variam linearmente com o parâmetro λ. Qualquer ciclo com custo reduzido não-negativo em λ' e λ'' tem custo reduzido não-negativo no intervalo $[\lambda', \lambda'']$. Desta forma, temos que $G(\mathcal{M})$ não possui ciclos de custo negativo sendo portanto o mparelhamento $\mathcal{M}$ ótimo no intervalo $[\lambda'', \lambda'']$. □

Lema 5.5.2 *Se para $\lambda = \lambda''$ existe um ciclo de custo zero $\mathcal{C}$ em $G(\mathcal{M})$ com custo decrescente com o parâmetro então λ'' é um ponto-de-quebra*

Prova: Para valores de λ maiores que λ'' o ciclo terá custo negativo. A partir de λ'' um outro emparelhamento ótimo existe. O valor λ'' caracteriza o término da otimalidade do emparelhamento $\mathcal{M}$ sendo portanto um ponto-de-quebra. □

Lema 5.5.3 *Seja $\mathcal{M}^*$ o emparelhamento obtido de $\mathcal{M}$ definindo as arestas emparelhadas presentes no ciclo de custo zero $\mathcal{C}$ como não-emparelhadas e as demais arestas de $\mathcal{C}$ como emparelhadas. O emparelhamento $\mathcal{M}^*$ é ótimo no intervalo $[\lambda'', \lambda'' + \varepsilon]$ para um $\varepsilon \geq 0$.*

Prova:O ciclo $\mathcal{C}$ possui custo zero de maneira que $\mathcal{M}$ e $\mathcal{M}^*$ possuem o mesmo custo. Temos assim que tanto $\mathcal{M}$ quanto $\mathcal{M}^*$ são ótimos em λ''. O emparelhamento $\mathcal{M}^*$ permanecerá ótimo quando variarmos o parâmetro enquanto não houver ciclos de custo negativo. Se existe um ciclo de custo zero em $G(\mathcal{M}^*)$ cujo custo decresce com o parâmetro, temos que $\mathcal{M}^*$ será ótimo apenas para o ponto λ''. Se não existem ciclos de custo zero

em $G(\mathcal{M}^*)$ cujo custo decresce com λ, $\mathcal{M}^*$ é ótimo no intervalo $[\lambda'', \lambda'' + \epsilon]$ para um específico $\epsilon > 0$. Se nenhum dos ciclos presentes em $G(\mathcal{M})$ possui custo decrescente com o parâmetro, o emparelhamento $\mathcal{M}^*$ é ótimo no intervalo $[\lambda'', \infty]$. □

A partir destes teoremas podem-se formular algoritmos para a resolução do PD paramétrico.

Um algoritmo para o PD paramétrico consiste em obter uma primeira solução, resolvendo uma instância do problema não-paramétrico, para o extremo inferior do intervalo estipulado para o parâmetro. A partir dessa primeira solução se determina em que momento irá surgir um ciclo de custo zero no grafo residual associado à esta solução. Baseado no ciclo de custo zero obtém-se um novo emparelhamento a partir do emparelhamento atual. O algoritmo prossegue avaliando a otimalidade do novo emparelhamento. O término do algoritmo é indicado quando todo o intervalo estipulado for varrido ou quando o último emparelhamento encontrado permanecer ótimo para $\lambda = \infty$.

Para finalizar a descrição deste algoritmo resta definir explicitamente cada operação. Por exemplo, como determinar o primeiro ciclo de custo zero que surge no grafo quando variamos o parâmetro? Verificou-se que determinar o primeiro ciclo de custo zero que surge no grafo residual está intimamente relacionado à resolução de uma instância do *problema do ciclo de razão mínima.*

5.5.1 O problema do ciclo de razão mínima

Seja $G' = (V', E')$ um grafo orientado. A cada aresta $e' \in E'$ estão associados valores inteiros $c_{e'}$ e $h_{e'}$. Referimos a estes valores como sendo respectivamente, o custo e o bônus da aresta. Existe uma variedade de problemas interessantes na literatura relacionado à instâncias com essa estrutura[3].

Para cada ciclo orientado C em G' definimos o seu custo como sendo a somatória dos custos das arestas do ciclo. Similarmente, pode-se definir o conceito de bônus para um ciclo orientado. O **problema do ciclo de razão mínima**, denotado por PCRM, consiste em determinar o ciclo orientado com a menor razão entre seu custo e seu bônus:

$$\varphi(C) = \frac{\sum_{e' \in C} c_{e'}}{\sum_{e' \in C} h_{e'}}$$

$$\varphi(\mathcal{W}) = \varpi = \min_{\forall C} \varphi(C)$$

Suporemos que $\sum_{e' \in C} h_{e'} > 0$ para todos os ciclos dirigidos de G. Desta forma todos os ciclos possuem razões bem-definidas. Denotaremos o ciclo de razão mínima desta instância

[3]Uma listagem destes problemas pode ser obtida no trabalho de Megiddo [Meg79]

por $\mathcal{W}$ e a razão deste ciclo por ϖ. O maior custo e o maior bônus das arestas da instância serão denotados, respectivamente, por $C = \max_{e' \in E'}\{c_{e'}\}$ e $H = \max_{e' \in E'}\{h_{e'}\}$.

O PCRM foi proposto por Lawler [Law67] e Dantzig e outros [DBR67]. Uma das primeiras aplicações do problema foi o estabelecimento de rotas de viagens ótimas. A literatura para o PCRM é relativamente restrita comparada ao seu caso especial mais importante, o problema do ciclo de média mínima-PCMM (seção 4.1.1). O PCMM é um caso específico do PCRM onde todas as arestas possuem bônus unitário. Ao contrário do PCMM que está envolvido na resolução de diversos problemas combinatórios, o PCRM é freqüentemente descrito na literatura no contexto do *problema tramp steamer* [Law73, Law76].

Wallacher e Zimmerman [WZ91] foram um dos poucos a aplicar ciclos de razão mínima em outros tópicos. Eles desenvolveram um algoritmo para o PFCM baseado no cancelamento de ciclos de razão mínima. Este algoritmo é um dos três algoritmos de tempo polinomial para o PFCM baseado em cancelamento de ciclos.

Considere o grafo G' obtido de uma instância G_λ do PD paramétrico definindo os custos como $c_{e'} = c^\bullet_e$ e os bônus $h_{e'} = c_e$. Para perceber porque o surgimento do primeiro ciclo de custo zero no grafo residual associado à um emparelhamento ótimo $\mathcal{M}'$ está intimamente relacionado ao PCRM, observe que o custo de um ciclo em G' representa a taxa de crescimento do custo $c^\bullet_e$ do ciclo correspondente em G em relação ao parâmetro λ. Não é difícil observar que o ciclo procurado deve ser um dos ciclos negativos de G'. Os ciclos de G associados à ciclos de custo positivo em G' jamais se tornam ciclos de custo zero.

O custo do ciclo procurado está estritamente decrescendo com o aumento do parâmetro. Dentre os vários ciclos cujos custos decrescem, o ciclo procurado consiste naquele cujo custo decai à zero primeiro. Este ciclo tem a menor razão entre o custo atual do ciclo e a taxa de decrescimento. Determinar tal ciclo consiste na resolução do problema:

$$\min_{\forall C \ \varphi(C)<0} \frac{\sum_{e \in C} c_e}{|\sum_{e \in C} c^\bullet_e|}$$

Este problema também pode ser entendido em relação a G' como:

$$\varpi = \min_{\mathcal{C}} \frac{\sum_{e' \in C} c^\bullet_{e'}}{\sum_{e' \in C} c_{e'}}$$

O último problema claramente se resume na resolução de uma instância do PCRM. O ciclo de custo zero procurado na resolução do PD paramétrico é o ciclo $\mathcal{W}$ de razão mínima em G'. O valor do parâmetro para o qual G conterá um ciclo de custo zero é $\lambda'' = \lambda' + |\frac{1}{\varpi}|$. Para o intervalo $[\lambda', \lambda' + |\frac{1}{\varpi}|]$ o emparelhamento $\mathcal{M}'$ é ótimo.

5.5.2 Tratando situações excepcionais

Na descrição do problema do ciclo de razão mínima, consideramos que o bônus do ciclo no grafo é positivo. Esta suposição não é satisfeita na abordagem do PD paramétrico proposta.

O bônus de cada aresta na instância do PCRM, gerada na resolução do PD paramétrico, é o custo da aresta no grafo residual da instância do PD. Assim sendo, não se poderá evitar a presença de ciclos de custo zero em $G'(\mathcal{M})$. Consequentemente, as instâncias do PCRM geradas não necessariamente se enquadram nas suposições feitas.

A razão para um ciclo $\mathcal{C}$ com bônus zero não é bem-definida ($\varphi(\mathcal{C}) = \pm\infty$). Os algoritmos para o PCRM podem ser aplicados em instâncias onde a suposição sobre os bônus não são satisfeitas. A solução nestes casos pode ser um ciclo com razão bem-definida ou não. Se a abordagem com ciclos de razão mínima é aplicada nas instâncias do PD paramétrico todos os pontos-de-quebra são corretamente encontrados.

Se existem vários ciclos com custo negativo e bônus zero ($\varphi = -\infty$) a solução encontrada na resolução do PCRM é um destes ciclos. Para cada um destes ciclos existirá um emparelhamento $\mathcal{M}''$ distinto. Dentre estes vários emparelhamentos apenas um é ótimo no intervalo $[\lambda'', \lambda'' + \varepsilon]$ para um $\varepsilon > 0$. Todos os demais emparelhamentos são ótimos apenas para o ponto λ'' sendo portanto desnecessários para a descrição das soluções ótimas de um PD paramétrico.

A existência de ciclos sem razão bem-definida não impede que o algoritmo para o PD paramétrico encontre todos os pontos-de-quebra corretamente. Entretanto, a presença destes ciclos pode provocar que o algoritmo compute várias instâncias do PCRM para encontrar um único ponto-de-quebra. Um limite para o número de instâncias resolvidas até a determinação de um ponto-de-quebra não é conhecido. A abordagem com ciclos de razão mínima como apresentada não atinge os objetivos de eficiência pretendidos. Para resolver o PD paramétrico apenas as soluções realmente imprescindíveis na descrição de $c_{\mathcal{M}}(\lambda)$ devem ser computadas.

Dentre os vários emparelhamentos ótimos existentes para o parâmetro λ'', deseja-se encontrar um emparelhamento $\mathcal{M}^*$ que é ótimo para um intervalo $[\lambda'', \lambda'' + \varepsilon]$ sendo $\varepsilon > 0$. Para λ'' $G(\mathcal{M}^*)$ não tem ciclos de custo zero. Devido à suposição de integralidade dos custos, um ciclo em $G(\mathcal{M}^*)$ deve ter custo no mínimo unitário. A maior taxa de decrescimento possível para um ciclo na instância do PD paramétrico é $nC^\bullet$. Desta maneira, sabemos que para o intervalo $[\lambda'', \lambda'' + \frac{1}{nC^\bullet}]$ o emparelhamento $\mathcal{M}^*$ procurado é ótimo. Este emparelhamento pode ser computado resolvendo uma instância do PD para λ no intervalo $[\lambda'', \lambda'' + \frac{1}{nC^\bullet}]$.

Uma abordagem um pouco melhor pode ser aplicada. Ao invés de escolher um valor qualquer no intervalo $[\lambda'', \lambda'' + \frac{1}{nC^\bullet}]$ poderia se escolher um valor racional no qual o emparelhamento $\mathcal{M}^*$ é ótimo sob ϵ para $\epsilon < \frac{\alpha}{n}$, onde α é uma constante inteira utilizada

como fator de escalonamento pelo algoritmo (seção 4.3.1). Neste caso, o emparelhamento ótimo $\mathcal{M}^*$ pode ser calculado resolvendo um único refinamento aplicando o algoritmo de escalonamento de custos. A complexidade na obtenção do emparelhamento ótimo seria reduzida para a complexidade de um refinamento $O(\sqrt{n}m)$.

Para determinar o valor do parâmetro para o qual um emparelhamento $\mathcal{M}^*$ é ótimo sob ϵ, necessita-se conhecer os potenciais dos vértices para o qual $\mathcal{M}^*$ é ótimo em λ''. Em seguida, estabelece-se o intervalo do parâmetro para o qual cada aresta não viola a condição de otimalidade de folga ϵ complementar por um valor superior a α. O valor racional escolhido deve estar presente no intervalo ótimo sob $\epsilon = \alpha$ de todas as arestas.

5.5.3 Algoritmos para o problema do ciclo de razão mínima

O primeiro algoritmo proposto para o PCRM é devido a Lawler [Law67]. Dantzig e outros [DBR67] apresentaram uma aproximação baseada na formulação do PCRM como um problema de Programação Linear.

Um método genérico para ser aplicado aos problemas de razão mínima foi proposto no trabalho de Megiddo [Meg79]. O principal resultado deste trabalho declara que algoritmos de tempo polinomial forte podem ser obtidos para um problema de razão mínima se existem algoritmos de tempo polinomial forte para o problema padrão relacionado.

Para a maioria dos algoritmos de Fluxos em Redes o teorema de Megiddo pode ser aplicado para a obtenção de soluções eficientes para os problemas de razão mínima relacionados. Especificamente para o caso do PCRM, o trabalho de Megiddo obtém o melhor limite de tempo polinomial forte $O(n^2 m \log n)$ para o problema.

Para o ciclo de razão mínima $\mathcal{W}$ temos que:

$$\varpi = \frac{\sum_{e \in \mathcal{W}} c_e}{\sum_{e \in \mathcal{W}} h_e}$$

Alternativamente, podemos declarar a expressão anterior como $\sum_{e \in \mathcal{W}} (c_e - \varpi h_e) = 0$. Observe que para valores superiores à ϖ o ciclo $\mathcal{W}$ possui custo negativo. Para todos os demais ciclos C temos que

$$\varpi < \varphi(C) = \frac{\sum_{e \in C} c_e}{\sum_{e \in C} h_e}$$

De outra forma, podemos verificar que $\sum_{e \in C} (c_e - \varpi h_e) > 0$.

Uma maneira de resolver o PCRM consiste em reduzi-lo ao PCMC paramétrico com arestas de custos $c_e - \Lambda h_e$. O ciclo de razão mínima pode ser determinado estabelecendo o maior valor do parâmetro Λ^{ϖ} para o qual os caminhos mais curtos são bem-definidos. O ciclo procurado é o ciclo de custo zero presente em $G_{\Lambda^{\varpi}}$ e a sua razão é Λ^{ϖ}.

Ao invés de aplicarmos o algoritmo desenvolvido por Karp e Orlin à instância do PCMC paramétrico utilizaremos uma abordagem mais simples, o algoritmo de Lawler. A

aproximação utilizada consistirá em gerar uma sequência de valores do parâmetro que convirja após um número finito de iterações a Λ^{ϖ}. Uma metodologia para gerar estas seqüências consiste em aplicarmos técnicas de busca.

Se limites inferior e superior são conhecidos tais que Λ^{ϖ} está presente no intervalo $[\Lambda^{+}, \Lambda^{\times}]$, a técnica de busca binária pode ser aplicada. Na *busca binária* a instância para o valor do parâmetro $\Lambda^{\varphi} = \frac{\Lambda^{+}+\Lambda^{\times}}{2}$ é avaliada. Se existe um ciclo de custo negativo no grafo, o valor do parâmetro testado é um limite superior para Λ^{ϖ}. Se os caminhos são bem-definidos mas nenhum ciclo de custo zero está presente na instância, temos que o valor testado é um limite inferior para Λ^{ϖ}. Por último, se um ciclo de custo zero é encontrado então este ciclo é o ciclo de razão mínima procurado.

Nesta aproximação o intervalo é continuamente reduzido até que o ciclo de razão mínima seja encontrado ou o intervalo se torne suficientemente pequeno de tal maneira que contenha uma única razão. A resolução de uma única instância dentro do intervalo é capaz de determinar o ciclo de razão mínima. O ciclo $\mathcal{W}$ é o ciclo de custo zero ou o ciclo negativo encontrado na avaliação desta última instância.

O número máximo de arestas em um ciclo é limitado a n, de maneira que a razão de qualquer ciclo está contida no intervalo $[-nC, nC]$. Considere os ciclos $\mathcal{W}$ e $\mathcal{C}$ com razões distintas. A diferença entre as razões dos ciclos é

$$\left|\frac{c(\mathcal{W})}{h(\mathcal{W})} - \frac{c(\mathcal{C})}{h(\mathcal{C})}\right| \neq 0 \qquad \left|\frac{c(\mathcal{W})h(\mathcal{C}) - c(\mathcal{C})h(\mathcal{W})}{h(\mathcal{W})h(\mathcal{C})}\right| \neq 0$$

A diferença entre as razões de $\mathcal{W}$ e $\mathcal{C}$ implica na expressão $c(\mathcal{W})h(\mathcal{C}) - c(\mathcal{C})h(\mathcal{W}) \geq 1$. Para a outra expressão temos que o maior valor é limitado a $h(\mathcal{W})h(\mathcal{C}) \leq H^2$. Concluímos assim que a diferença entre a razão de dois ciclos distintos é no mínimo $\frac{1}{H^2}$. Se o intervalo contendo Λ^{ϖ} reduz-se a uma faixa inferior a $[\Lambda^{+}, \Lambda^{\times}] < \frac{1}{H^2}$, o algoritmo pode terminar a busca porque este intervalo contém no máximo uma razão.

A complexidade da aproximação depende do número de iterações e da complexidade de cada iteração. No início o intervalo é $[\Lambda^{+}, \Lambda^{\times}] = 2nC$. No mínimo $O(\log(HC))$ iterações são necessárias para o intervalo se reduzir a $\frac{1}{H^2}$. Cada iteração envolve a resolução de uma instancia do PCMC. Utilizando o algoritmo de escalonamento de custos de Goldberg e Tarjan para o PD para a computação dos caminhos mais curtos, o algoritmo de Lawler pode ser implementado em tempo polinomial fraco $O(\sqrt{n}m \log(nC) \log(HC))$.

O algoritmo de escalonamento de custos, por ser um algoritmo com complexidade polinomial fraca, não pode ser aplicado à instâncias com custos não integrais. Para aplicarmos o algoritmo de escalonamento de custos como rotina para a detecção de ciclos negativos, na abordagem com ciclos de razão mínima, temos que garantir a integralidade dos custos das arestas. Durante a busca binária, o valor de Λ pode se tornar não integral. Este problema pode ser contornado se modificarmos os custos das arestas para

$$c_e^* = H^2 c_e - \Lambda h_e$$

Com esta modificação a razão dos ciclo do grafo foi multiplicada por H^2. Esta modificação não altera o ciclo de razão mínima. Se definirmos o intervalo de busca como sendo $[-nCH^2, nCH^2]$, o algoritmo de Lawler pode terminar a busca quando o intervalo for $[\Lambda^+, \Lambda^\times] \leq 1$.

Outro aspecto que deve ser tratado é a integralidade dos pontos-de-quebra. Os bônus das arestas nas instâncias do PCRM estão relacionadas aos custos da instância do PD paramétrico, o que significa que os bônus dependem do parâmetro λ. Supomos desde o princípio que os custos das arestas numa instância do PD paramétrico são formadas for uma função linear com coeficientes integrais. Na resolução do PD paramétrico, o primeiro ponto-de-quebra a ser computado será um número racional (razão entre números inteiros). Os pontos-de-quebra subsequentes são também números racionais, visto que são a razão entre número racionais. Desta forma, temos que todos os pontos-de-quebra são racionais. Os números racionais podem ser integralizados multiplicando-os por constantes inteiras suficientemente grandes.

O algoritmo de escalonamento de custo se aplica à instâncias com dados racionais. Desta maneira, a não integralidade dos pontos-de-quebra também pode ser adequadamente tratada. Utilizando o algoritmo de escalonamento de custos na implementação da abordagem com ciclos de razão mínima para o PD paramétrico, cada ponto-de-quebra pode ser computado com complexidade $O(\sqrt{n}m \log(nC) \log(HC) + \sqrt{n}m)$.

Aplicando o algoritmo de Megiddo para o PCRM e o algoritmo Primal-dual para o PD, a abordagem com ciclos de razão mínima possui complexidade polinomial forte $O(n^2 m \log n + nm + n^2 \log n)$.

5.6 Estendendo a abordagem ao problema do fluxo de custo mínimo paramétrico

A abordagem com ciclos de razão mínima permite que cada solução do PD paramétrico seja obtida em tempo $O(\min\{\sqrt{n}m \log(nC) \log(HC) + \sqrt{n}m, nm^2 \log n\})$. Comparando esta abordagem ao trabalho de Eisner e Severance observamos que nenhum melhoramento para o problema foi conseguido. O principal motivo para a ineficiência da abordagem está no fato do PCRM ser um problema mais complexo para se resolver que o próprio PD.

As condições de otimalidade assim como os algoritmos aplicados ao PD são especializações dos conceitos desenvolvidos para o problema mais genérico, o PFCM. Estes conceitos são adaptados ao contexto do PD para explorar eficientemente a sua estrutura peculiar. Sendo assim, podem-se estudar as idéias na abordagem com ciclos de razão mínima no contexto do PFCM paramétrico. Determinar os valores do parâmetro para o qual um determinado fluxo permanece ótimo envolve determinar o intervalo para o qual

o grafo residual associado à este fluxo permanece sem ciclos de custo negativo. Este problema, mesmo no contexto do PFCM paramétrico, recai na resolução de uma instância do problema do ciclo de razão mínima.

As iterações do algoritmo discutidas para o PD paramétrico quando aplicadas ao PFCM paramétrico permanecem com a mesma complexidade com exceção da obtenção da nova solução. Em ambos os casos aumenta-se o fluxo o máximo possível através do ciclo de razão mínima. No contexto do PD, o aumento do fluxo é similar a trocar as arestas emparelhadas do ciclo com as não-emparelhadas. Devido à presença dos ciclos sem razão bem-definida, para melhorar a complexidade do processo pode-se utilizar a solução atual como uma boa aproximação para a próxima solução. Sabendo-se que a nova solução é ótima no mínimo para o intervalo $[\lambda'', \lambda'' + |\frac{1}{nC^*}|]$, pode-se buscar um valor racional do parâmetro no intervalo para o qual a solução é ótima sob $\epsilon < \frac{\alpha}{n}$, onde α é o fator de escalonamento. Aplicando o algoritmo de escalonamento de custos a nova solução pode ser encontrada com um único refinamento. No caso do PFCM a complexidade de cada refinamento é $O(nm \log \frac{n^2}{m})$.

Concluímos deste modo que a abordagem com ciclos de razão mínima pode ser estendida com poucas modificações ao PFCM paramétrico. A complexidade da abordagem com ciclos de razão mínima para o PFCM é $O(\sqrt{n}m \log(nC) \log(HC) + nm \log \frac{n^2}{m})$ ou polinomial forte usando o algoritmo de Orlin [Orl88] para o PFCM e o algoritmo de Megiddo $O(n^2 m \log n + (m \log n)(m + n \log n))$.

Comparando a complexidade dos algoritmos para o PFCM (tabela 4.1) e para o PCRM (seção 5.5.3), vemos que estes problemas podem ser resolvidos praticamente no mesmo tempo. A diferença entre os algoritmos para instâncias de mesmo tamanho e mesma densidade não é superior a $O(n)$. Considerando que a hipótese de similaridade (seção 1.2) é satisfeita, o melhor limite de tempo para o PFCM é $O(nm \log \frac{n^2}{m} \log(nC))$ e para o PCRM é $O(\sqrt{n}m \log(nC) \log(HC))$. A abordagem com ciclos de razão mínima ao PFCM paramétrico obterá cada solução do problema por um fator $O(\log(nC))$ menor do que o tempo de resolução de uma instância do PFCM.

Se a hipótese de similaridade não é satisfeita, a abordagem proposta não oferece melhoramentos em relação a solução desenvolvida por Eisner e Severance. A aplicação da abordagem com ciclos de razão mínima permite obter para o PFCM paramétrico, pelos menos em casos específicos, o resultado que buscávamos para o PD paramétrico.

A abordagem com ciclos de razão mínima no nosso entendimento é a primeira abordagem a garantir que a estrutura de cada solução ótima distinta do conjunto de soluções ótimas do PFCM paramétrico pode ser obtida num tempo inferior à resolução de uma instância do PFCM. Para isto nós impusemos que os custos das arestas são inteiros e limitados polinomialmente.

Limites para o número máximo de soluções para cada um destes problemas paramétri-

cos não são conhecidos. Se analisarmos a estrutura das soluções geradas no PD paramétrico e no PFCM paramétrico observaremos que os problemas diferem também no número máximo de soluções possíveis. O grafo residual de cada nova solução obtida para o PD paramétrico difere do grafo residual da anterior no mínimo em um ciclo, o ciclo de razão mínima. No caso do PFCM paramétrico, os grafos residuais de fluxos consecutivos diferem no mínimo em uma aresta. Esta aresta é a aresta que se tornou saturada no ciclo de razão mínima com o aumento do fluxo.

Devido ao maior grau de modificação, o PD paramétrico tende a convergir mais rapidamente para a última solução ótima do problema. Assim como os métodos de cancelamento de ciclos e outros, os mesmos conceitos e algoritmos podem ser aplicados ao PD e ao PFCM. Entretanto, a estrutura combinatória de cada problema lhes conferirá uma complexidade distinta.

Capítulo 6

Implementações

Este capítulo descreve o estudo experimental realizado sobre as duas únicas abordagens específicas propostas para o *problema do fluxo de custo mínimo paramétrico* com custos das arestas parametrizados. A primeira abordagem é a solução proposta por Srinivasan e Thompson, discutida na seção 5.3. A segunda abordagem específica é a solução proposta nesta dissertação, descrita na seção 5.5, que se refere à computação de ciclos de razão mínima.

A abordagem com ciclos de razão mínima explora a estrutura do PFCM para computar eficientemente cada um dos fluxos ótimos do problema paramétrico. A principal vantagem desta abordagem é a computação dos fluxos ótimos em tempo inferior à complexidade de resolução de uma instância do PFCM, se a hipótese de similaridade é satisfeita (seção 1.2).

A respeito da abordagem de Srinivasan e Thompson nenhuma análise sobre a sua complexidade computacional é conhecida. Em termos teóricos, a abordagem com ciclos de razão mínima provê os melhores resultados computacionais para o PFCM paramétrico. O objetivo deste estudo é analisar o comportamento empírico das duas abordagens específicas e avaliar se a abordagem com ciclos de razão mínima provê também melhoramentos práticos na resolução do problema.

6.1 Geração de instâncias

O objetivo deste estudo experimental é avaliar qual é a melhor abordagem para o PFCM paramétrico e se as características das instâncias influenciam o comportamento dos algoritmos. Para isto as implementações realizadas foram testadas em diversas classes de instâncias. Uma ferramenta fundamental na realização deste estudo foi a automatização do processo de geração de instâncias. Diversos geradores foram aplicados na construção das 8 famílias de instâncias avaliadas.

Uma característica comum a todos os geradores utilizados é a aplicação de rotinas de

números pseudo-aleatórios para produzir as informações do problema, como por exemplo, os custos das arestas. Esta característica permite que uma determinada instância possa ser gerada novamente se os mesmos parâmetros são aplicados e também obter diversas instâncias distintas para uma mesma estrutura.

A geração das instâncias paramétricas foi efetuada em duas fases. Em uma primeira fase, foram geradas instâncias não parametrizadas do PFCM com diversas estruturas específicas. Em uma fase posterior cada instância foi parametrizada.

Para a geração das instâncias não parametrizadas utilizaram-se os geradores *Netgen*, *Grid-on-torus* e *Grid-graph* de domínio público do *DIMACS - Center for Discrete Mathematics and Theoretical Computer Science*[1]. Para realizar a segunda fase no processo de construção das instâncias, foi implementado um gerador denominado *Grcp* cuja a finalidade é parametrizar instâncias.

A descrição de todos os geradores utilizados é feita no apêndice A.

6.2 Implementação baseada no algoritmo simplex de rede

Para a implementação da abordagem do algoritmo de Srinivasan e Thompson, para o PFCM paramétrico com custos parametrizados, foi utilizado a implementação do algoritmo simplex de rede de A. V. Goldberg. O código denominado SI está implementado em linguagem C[2].

Por não haver informações do autor sobre o desempenho do SI em relação aos demais códigos existentes para o PFCM, realizou-se um estudo experimental. Neste estudo experimental analisou-se além do SI, o CS2 e o NETFLO. O CS2 desenvolvido por Goldberg é uma implementação do algoritmo de escalonamento de custos de Goldberg e Tarjan [GT88]. O clássico NETFLO é uma implementação de Kennington e Helgason [KH80] do algoritmo simplex de rede. O estudo realizado nos moldes do trabalho de Goldberg [Gol97] indica a superioridade total do SI sobre o NETFLO. Em relação ao CS2, o SI tende a ser mais lento na maioria das classes avaliadas para instâncias suficientemente grandes. Em classes específicas o desempenho dos algoritmos foi equivalente. Os tempos de execução foram diferentes por um fator constante. Numa única classe o desempenho do SI foi superior ao do CS2.

O SI será executado primeiramente para obter o primeiro fluxo ótimo $\mathcal{F}$ e a correspondente árvore espalhada $\mathcal{T}$. Os próximos fluxos serão computados com base nas soluções

[1] A documentação e os códigos fontes destes geradores podem ser obtidos eletronicamente via ftp em ftp://dimacs.rutgers.edu

[2] Este código não está em domínio público e foi gentilmente cedido exclusivamente para os fins deste estudo experimental

obtidas. Para implementar o algoritmo de Srinivasan e Thompson algumas mudanças na estrutura de dados do código foram necessárias para armazenar informações da variante paramétrica do problema. As alterações incluíram também a inserção de uma rotina no código fonte do SI para implementar os testes discutidos na seção 5.3 sobre a otimalidade de uma solução de árvore espalhada.

A rotina inserida primeiramente obtém os potenciais dos vértices para o qual a árvore T° associada a um fluxo $\mathcal{F}_{\lambda^\circ}$ é ótima. Os potenciais são obtidos percorrendo a árvore espalhada em pré-ordem. Calcula-se para cada aresta fora da árvore espalhada o maior valor do parâmetro para o qual a condição de otimalidade de custo reduzido não é violada. Determina-se em seguida a primeira aresta $e \notin T^\circ$ a violar a condição de otimalidade e o maior valor do parâmetro λ^* para o qual a aresta permanece sendo ótima.

Em seguida, calculam-se os valores das arestas e os potenciais ótimos para a árvore T_{λ° para o valor do parâmetro igual a $\lambda = \lambda^*$. Uma rotina do SI é invocada para realizar um pivotamento com e como sendo a aresta de entrada. Com o pivotamento uma nova árvore espalhada ótima T^* é obtida. Se o pivotamento realizado não é degenerado o fluxo ótimo $\mathcal{F}_{\lambda^*}$ associado a nova árvore difere do anterior. Se $\lambda^\circ \neq \lambda^*$ um novo ponto-de-quebra foi encontrado. Neste caso as informações relativas à nova solução encontrada são impressas. A execução prossegue determinando os fluxos subsequentes até que a árvore espalhada permaneça ótima para todos os valores do parâmetros subsequentes.

6.3 Implementação baseada em ciclos de razão mínima

Na implementação da abordagem com ciclos de razão mínima dois códigos denominados CS2 e CSA foram combinados. Ambos os códigos implementam o algoritmo de escalonamento de custos. O CS2 desenvolvido por Goldberg implementa o algoritmo de escalonamento de custos no contexto do PFCM e o CSA desenvolvido por Goldberg e Kennedy implementa o algoritmo no contexto do PD[3].

Na implementação realizada o CS2 foi utilizado para obter o primeiro fluxo ótimo para o problema para $\underline{\lambda} = 0$. A estrutura de dados do CS2 foi alterada com a inclusão de campos para o armazenamento das informações sobre a variação do custo das arestas em relação ao parâmetro. As alterações incluíram também modificações no código fonte do CS2.

O CSA foi utilizado como uma rotina para a detecção de ciclos de custo negativo como discutido na seção 3.4. A detecção de ciclos negativos é necessária para a implementação

[3]Ambos os códigos em linguagem C podem ser obtidos gratuitamente na seguinte URL http://www.intertrust.com/star-lab.com/goldberg/

do algoritmo de Lawler para o problema do ciclo de razão mínima. A implementação do algoritmo de Lawler realizada foi inserida no próprio código do CSA.

Após a computação do primeiro fluxo ótimo, o CS2 inicializa as estruturas de dados do CSA. Em seguida, o CSA é invocado para computar o primeiro ciclo de razão mínima no grafo residual do fluxo corrente, como discutido na seção 5.5. Se a razão do ciclo de razão mínima retornada pelo CSA é negativa então existe um ponto-de-quebra estritamente maior que o valor atual do parâmetro. Neste caso as informações sobre o novo fluxo obtido são impressas. Caso contrário, todos os fluxos ótimos foram computados e o CS2 finaliza a execução imprimindo as últimas informações.

No código do CSA primeiramente se avalia se o fluxo corrente $\mathcal{F}$ ótimo para $\underline{\lambda}$ é ótimo também para alguns valores do parâmetro subsequentes. Nesta parte da implementação optou-se por um algoritmo diferente da proposta na abordagem com ciclos de razão mínima. Ao invés de aplicar uma fase escalonada com o algoritmo de escalonamento de custos para obter o novo fluxo ótimo no intervalo $[\underline{\lambda}, \underline{\lambda} + \epsilon]$ para um $\epsilon > 0$, o novo fluxo é obtido através de um processo de cancelamento dos ciclos de custo zero do grafo residual. Recorde que o grafo residual do fluxo procurado não possui ciclos de custo zero com custo decrescente com o parâmetro.

Optou-se por esta variação por ser mais simples, em somente algumas iterações têm-se grafos residuais sem ciclos com razão bem-definida. A obtenção do novo fluxo a partir do cancelamento dos ciclos em algumas iterações se torna mais penoso do que aplicar uma fase escalonada do algoritmo de escalonamento de custos. Entretanto, quando não é necessário por existirem apenas ciclos com razão bem-definida gasta-se muito menos tempo nesta operação. Da forma como foi implementado não se pode garantir que a execução do algoritmo será em tempo polinomial. A opção por este implementação somente se justifica pelo fato de que tentou-se implementar o algoritmo da maneira mais conveniente. Poderia ter sido investido mais tempo na implementação do algoritmo, mas acreditamos que os testes realizados foram suficientes para obter uma conclusão confiável do comportamento da abordagem com ciclos de razão mínima. Não se espera que esta implementação seja a melhor entre todas as possíveis para a abordagem com ciclos de razão mínima.

O código implementado avalia se existem ciclos de custo zero em $G(\mathcal{F})$ cujo custo decresce com o parâmetro λ. Caso exista algum ciclo $\mathcal{C}$ com custo decrescente este é cancelado aumentando-se o fluxo o máximo possível nas arestas. Para encontrar o ciclo $\mathcal{C}$ gera-se um subgrafo contendo apenas as arestas com custo reduzido zero em $G(\mathcal{F})$. Substitui-se em seguida os custos das arestas do subgrafo pelos respectivos coeficientes c^*. O ciclo $\mathcal{C}$ se existir é um dos ciclos negativos no subgrafo.

A partir de $\mathcal{C}$ obtém-se um novo fluxo resultante. Este fluxo obtido é ótimo para $\underline{\lambda}$ mas não necessariamente é ótimo para valores maiores que $\underline{\lambda}$. A avaliação anterior é realizada continuamente até que se obtenha um fluxo que seja ótimo para valores maiores

que $\underline{\lambda}$. Após a obtenção do novo fluxo gera-se uma instância do PCRM para computar o ponto-de-quebra estritamente maior que $\underline{\lambda}$.

Para as arestas de custo $c_e = c_e^\circ + \lambda' c_e^\bullet$, a instância do PCRM gerado tem arestas com custos $c_{e'} = C^2 c_e^\bullet$ e bônus $h_{e'} = c_e$. O ciclo de razão mínima procurado se encontra no intervalo $[-nHC^2, nHC^2]$. Para determinar o ciclo de razão mínima o algoritmo de Lawler gera um grafo $\mathcal{G}$ com arestas de custo

$$c_{e'} = C^2 c_e^\bullet - \Lambda c_e$$

Para manter a integralidade dos custos o parâmetro Λ é representado por uma fração de dois números inteiros $\Lambda = \frac{\Lambda^\times}{\Lambda^+}$. A expressão que define o custo das arestas de $\mathcal{G}$ é redefinida como

$$c_{e'} = C^2 c_e^\bullet \Lambda^+ - \Lambda^\times c_e$$

Em seguida o algoritmo de Lawler é aplicado com uma variação. Se um ciclo negativo $\mathcal{C}$ é encontrado em Λ^* o próximo valor de Λ avaliado não será o ponto médio do intervalo de busca, mas a razão

$$\Lambda = \frac{h(\mathcal{C})}{c(\mathcal{C})}$$

onde $h(\mathcal{C})$ é o bônus do ciclo e $c(\mathcal{C})$ é o custo do ciclo em Λ^*.

A justificativa para esta variação está no fato de que para $\Lambda = \frac{h(\mathcal{C})}{c(\mathcal{C})}$ o ciclo $\mathcal{C}$ tem custo zero. No intervalo $[\Lambda^*, \frac{c(\mathcal{C})}{h(\mathcal{C})}]$ o ciclo $\mathcal{C}$ tem custo negativo e a razão do ciclo de razão mínima procurado não pode estar neste intervalo. Se nenhum ciclo de custo negativo é encontrado a iteração prossegue como descrito no algoritmo de Lawler avaliando os pontos médios do intervalo de busca.

A primeira iteração do algoritmo de Lawler quando $\Lambda = 0$ estabelece se o fluxo corrente permanecerá ótimo indefinidamente ou não. Se existe um ciclo de custo negativo no grafo para $\Lambda = 0$ nenhum dos ciclos no grafo residual tem custo que decresce com o parâmetro. Neste caso todos os fluxos ótimos foram computados. Caso contrário, mais iterações são realizadas até a determinação do ciclo de razão mínima. Por último a implementação do algoritmo de Lawler retorna a razão do ciclo de razão mínima encontrado.

6.4 Classes do problema avaliadas

Os três geradores do *DIMACS* foram utilizados para a construção de 8 famílias de instâncias do PFCM. Cada família por sua vez originou duas versões de problemas paramétricos. Na primeira versão a parametrização dos custos é definida sem restrições. O valor máximo permitido para o coeficientes dos parâmetros foi 50. Na segunda versão foi realizada uma parametrização 0-1. Ambas as implementações realizadas foram avaliadas em todas as versões de todas as famílias.

Famílias torus O *Grid-on-torus* foi utilizado para criar duas famílias de instâncias denominadas torus-$\sqrt[3]{n^5}$ e torus-$6n$. A diferença nestas duas famílias de instâncias está na densidade do grafo gerado. Na família torus-$\sqrt[3]{n^5}$ gerou-se as instâncias o mais denso possível permitido pelo gerador com $m = \sqrt[3]{n^5}$. Na família torus-$6n$ as instâncias são as mais esparsas possíveis. A quantidade de arestas é estabelecido com base no número de vértices como $m = 6n$. Os demais parâmetros de entradas são comuns a ambas as classes. A capacidade máxima para as arestas é definida como $U = 400$. O custo máximo permitido para uma aresta é $C = 100$. Nestas famílias foram geradas instâncias de tamanho 20, 40, 60 e 80 vértices.

Famílias Grid O *Grid-graph* foi utilizado para a construção de três famílias de instâncias denominadas grid-nxn, grid-hxn e grid-nxw. A diferença nestas três famílias está na estrutura da grade retangular gerada. Na família grid-nxn todas as instâncias tem a altura igual a largura da grade. Na família grid-hxn todas as instâncias tem a altura fixada em $h = 5$. Na família grid-nxw em todas as instâncias a largura é fixada em $w = 5$. Os demais parâmetros (custos e capacidades) das arestas são fixados em $C = U = 100$. Nestas famílias foram geradas instâncias com n sendo 7, 8, 9 e 10.

Famílias Netgen O *Netgen* foi utilizado para a construção de três famílias de instâncias denominadas netgen-one, netgen-quarter e netgen-medium. As três famílias se caracterizam por serem estruturalmente distintas. Na família netgen-one as instâncias geradas têm a estrutura típica do PFCM com uma super-fonte e um super-sorvedouro. Todos os demais vértices são vértices de entreposto. Na família netgen-quarter e netgen-medium as instâncias se caracterizam por terem diversas fontes e diversos sorvedouros. Na família netgen-quarter metade dos vértices são de entreposto. Dos vértices restantes metade são fonte e metade sorvedouros. Na família netgen-medium não existem vértices de entreposto. Os vértices são divididos igualmente entre fontes e sorvedouros. Nestas famílias foram geradas instâncias com k igual a 2, 4, 6 e 8.

Os demais parâmetros para as três classes são fixos ou definidos em função do número de vértices.

vértices	$n = 10k$
densidade	$m = 3\ 2^{\frac{k}{2}+5}$
custo mínimo	$\underline{c} = 0$
custo máximo	$\overline{c} = 100$
excesso	$b = 20\ 2^{\frac{k}{2}}$
fontes de entreposto	$n_{ts} = 0$
sorvedouros de entreposto	$n_{tt} = 0$

arestas no esqueleto[4] com custo máximo	$\%_c = 100$
arestas capacitadas	$\%_u = 100$
capacidade mínima	$\underline{u} = 1$
capacidade máxima	$\overline{u} = 400$

6.5 Informações sobre a aplicação dos testes

Os códigos da abordagem de Srinivasan e Thompson e da abordagem com ciclos de razão mínima foram escritos em linguagem C e compilados utilizando o compilador GNU-gcc com as seguintes opções de otimização -04 -g. Todos os estudos experimentais foram realizados numa máquina DIGITAL 600Au contendo 1 Gigabyte de memória. Os tempos de execução anotados são os tempos efetivos da aplicação do algoritmo não incluindo o tempo gasto em operações de entrada e saída. A precisão dos tempos é da ordem de $\frac{1}{60}$ segundos.

A máquina utilizada para a realização dos testes experimentais não esteve dedicada exclusivamente para a execução das 2 abordagens avaliadas. Devido aos tempos de execução baixos e a carga de trabalho na máquina os tempos medidos para a resolução de instâncias de mesmo tamanho sofreram variações. A metodologia escolhida para a avaliação dos códigos consistiu em avaliar 5 instâncias para cada tamanho. O tempo anotado nas tabelas é o tempo médio de resolução das 5 instâncias.

Na implementação com ciclos de razão mínima, os custos das arestas são convertidos para números inteiros para se aplicar o algoritmo de escalonamento de custos para o PD como rotina para a detecção de ciclos negativos. A conversão é feita multiplicando-se os custos por valores suficientemente grandes[5]. Na máquina utilizada neste estudo a precisão para os números de ponto-flutuante é limitada a 53 bits. Devido às restrições de precisão o tamanho das instâncias avaliadas foi reduzido. Os custos das arestas e os coeficientes do parâmetro também foram limitados para permitir a avaliação do algoritmo baseado em ciclos de razão mínima.

Apesar da limitação dos testes a instâncias de pequeno tamanho e custos limitados algumas instâncias em determinadas classes excederam a precisão disponível, não sendo o algoritmo capaz de finalizar a sua execução. A precisão máxima somente foi alcançada na execução das instâncias com parametrização genérica sem restrições quanto ao coeficiente $c^\bullet$ das arestas. Nestas instâncias os custos das arestas e das soluções alcançaram maiores valores.

[4] O *Netgen* gera primeiro um esqueleto um subgrafo básico em que ele garante a factibilidade do problema e outras características. Este subgrafo é então incrementado com as demais características para se encontrar um grafo com todas as características desejadas.

[5] O valor utilizado é o $\text{mmc}(\Lambda^+, \Lambda^\times)$

Os resultados dos testes realizados são expostos em tabelas. As tabelas exibem informações sobre as instâncias avaliadas, sobre a abordagem de Srinivasan e Thompson (ST) e sobre a abordagem com ciclos de razão mínima (CRM). Na porção da tabela referente às informações das instâncias tem se a informação sobre:

vértices número de vértices

soluções número mínimo e máximo de fluxos ótimos distintos obtidos na resolução das 5 instâncias deste tamanho.

Na porção da tabela referente a abordagem ST têm se informações sobre:

degeneração a maior porcentagem de pivotamentos degenerados (pivotamentos que não implicam na obtenção de novos fluxos) verificados na resolução das instâncias deste tamanho

desvio padrão desvio padrão das porcentagens de pivotamentos degenerados verificados nas instâncias de mesmo tamanho. Este campo indica se a degeneração se manteve numa porcentagem uniforme ou houve uma considerável diferença na degeneração entre instâncias de mesmo tamanho.

SI tempo de execução do SI para a obtenção do primeiro fluxo ótimo

tempo(s) tempo de execução total para a abordagem de Srinivasan e Thompson

Na parte final da tabela estão as informações referentes à abordagem com ciclos de razão mínima:

CS2 tempo de execução do CS2 para a obtenção do primeiro fluxo ótimo

tempo(s) tempo de execução total para a abordagem com ciclos de razão mínima

6.6 Resultados experimentais

Os melhores resultados para a abordagem com ciclos de razão mínima foram obtidos nas classes de instâncias geradas a partir do *Netgen*. Nestas classes os tempos de execução para a abordagem com ciclos de razão mínima ficaram mais próximos dos tempos obtidos com a abordagem de Srinivasan e Thompson.

O desempenho mais fraco da abordagem baseada no algoritmo simplex de rede se deve à degeneração apresentada na resolução das instâncias. Esta família de instâncias se caracterizou por apresentar uma grande degeneração independente do tipo de parametrização

aplicada. Nas três classes avaliadas o grau de degeneração se manteve superior à degeneração observada nas demais famílias. Na classe netgen-one se obteve os maiores índices de degeneração dentre todas as classes de instâncias avaliadas neste estudo. A degeneração se manteve sempre acima dos 80% chegando em algumas instâncias a apresentar 97%. Nas classes netgen-quarter e netgen-medium o grau de degeneração médio foi mais próximo do observado em outras classes mantendo-se em torno dos 50%.

A degeneração na classe netgen-one está relacionada principalmente à estrutura das instâncias geradas pelo *Netgen* e não pela presença de uma super-fonte e um super-sorvedouro. Nas famílias geradas pelo *Grid-on-torus* e pelo *Grid-graph* todas as instâncias possuem uma super-fonte e um super-sorvedouro sendo que mesmo assim a degeneração foi menor.

Na maioria das classes avaliadas a variação observada na degeneração entre instâncias de mesmo tamanho foi pequena ficando o desvio padrão inferior a 10. Na classe com maior variação observada o desvio padrão não foi superior a 17.

As classes da família Netgen foram as que apresentaram os resultados mais uniformes para instâncias de mesmo tamanho. O número de fluxos ótimos computados e a degeneração verificados na resolução destas instâncias variaram pouco independente da parametrização aplicada.

Nas classes netgen-quarter e netgen-medium os tempos de execução foram próximos para uma mesma abordagem. Nas tabelas 6.1, 6.2 e 6.3 são apresentados os tempos de execução obtidos para as instâncias com parametrização genérica nas três classes de instâncias.

Dentre todas as classes avaliadas o melhor desempenho da abordagem com ciclos de razão mínima ocorreu na classe netgen-one. Nesta classe os tempos de execução da abordagem com ciclos de razão mínima foram 4 vezes maiores que os tempos obtidos com a abordagem de Srinivasan e Thompson. Nas classes netgen-quarter e netgen-medium os tempos com a abordagem com ciclos de razão mínima foram em torno de 50 vezes maiores(veja figuras 6.1 e 6.2). Para as classes dos demais geradores a diferença entre os tempos de execução com a abordagem com ciclos de razão mínima esteve sempre superior a 50 vezes os tempos da abordagem de Srinivasan e Thompson.

O número de fluxos ótimos computados nas instâncias da classe netgen-one foi consideravelmente menor do que a quantidade de fluxos ótimos para as classes netgen-quarter e netgen-medium. Esta variação menor ocorreu independente da parametrização aplicada. Este aspecto reflete diretamente nos tempos de execução das instâncias da classe netgen-one que são visivelmente os menores para esta família. Na classe netgen-one uma instância com parametrização genérica de tamanho 80 excedeu a precisão máxima não sendo resolvida pela abordagem com ciclos de razão mínima.

Em todas as famílias avaliadas os tempos de execução foram menores nas instâncias

Classe netgen-one							
instância		*abordagem ST*				*abordagem CRM*	
vértices	*soluções*	*degeneração*	*d.padrão*	SI	*tempo(s)*	CS2	*tempo(s)*
20	3/7	87.8%	3.87	0	0	0	0.02
40	4/7	94.2%	1.33	0	0.02	0	0.04
60	5/12	94%	2.09	0	0.05	0	0.22
80	9/43	93.6%	3	0	0.24	0.01	0.69

Tabela 6.1: *Resultados obtidos para a classe netgen-one*

Classe netgen-quarter							
instância		*abordagem ST*				*abordagem CRM*	
vértices	*soluções*	*degeneração*	*d.padrão*	SI	*tempo(s)*	CS2	*tempo(s)*
20	18/34	50.8%	9.98	0	0	0	0.16
40	52/62	50.4%	3.38	0	0.03	0	1.23
60	91/101	52.4%	6.28	0	0.09	0	5.14
80	143/188	57.2%	2.48	0	0.40	0.01	17.14

Tabela 6.2: *Resultados obtidos para a classe netgen-quarter*

Classe netgen-medium							
instância		*abordagem ST*				*abordagem CRM*	
vértices	*soluções*	*degeneração*	*d.padrão*	SI	*tempo(s)*	CS2	*tempo(s)*
20	20/28	46.2%	1.16	0	0	0	0.13
40	45/65	53.2%	5.15	0	0.03	0	1.08
60	88/117	50.6%	3	0	0.09	0	4.85
80	182/204	48.2%	1.72	0	0.40	0.01	20.69

Tabela 6.3: *Resultados obtidos para a classe netgen-medium*

Classe netgen-one							
instância		*abordagem ST*				*abordagem CRM*	
vértices	*soluções*	*degeneração*	*d.padrão*	SI	*tempo(s)*	CS2	*tempo(s)*
20	1/3	86.2%	3.31	0	0	0	0.01
40	1/3	91.6%	3.26	0	0	0	0.01
60	1/7	91.8%	4.44	0	0.01	0	0.06
80	3/13	90.6%	3.20	0	0.02	0.01	0.30

Tabela 6.4: *Resultados obtidos para a classe netgen-one com parametrização 0-1*

Classe netgen-quarter							
instância		*abordagem ST*				*abordagem CRM*	
vértices	*soluções*	*degeneração*	*d.padrão*	SI	*tempo(s)*	CS2	*tempo(s)*
20	6/13	51.6%	9.35	0	0	0	0.04
40	9/23	55.4%	8.35	0	0.01	0	0.24
60	29/42	52.2%	6.88	0	0.03	0	1.20
80	31/40	66.4%	7.05	0	0.10	0.01	2.81

Tabela 6.5: *Resultados obtidos para a classe netgen-quarter com parametrização 0-1*

Classe netgen-medium							
instância		*abordagem ST*				*abordagem CRM*	
vértices	*soluções*	*degeneração*	*d.padrão*	SI	*tempo(s)*	CS2	*tempo(s)*
20	10/19	42.2%	10.44	0	0	0	0.05
40	15/21	53%	8.74	0	0.02	0	0.26
60	31/34	57%	5.37	0	0.03	0	1.23
80	44/59	59%	6.29	0	0.13	0.01	4.37

Tabela 6.6: *Resultados obtidos para a classe netgen-medium com parametrização 0-1*

com parametrização 0-1. A redução ocorreu principalmente devido ao número inferior de fluxos ótimos encontrados nas instâncias com parametrização 0-1. Na maioria das classes avaliadas a degeneração das instâncias com parametrização 0-1 seguiu o comportamento verificado nas instâncias da mesma classe com parametrização genérica.

Na classe netgen-one o desempenho da abordagem com ciclos de razão mínima para as instâncias com parametrização 0-1 foi pior em relação as instâncias com parametrização genérica. Os tempos de execução se mantiveram na ordem de 10 maiores que os tempos obtidos pela abordagem de Srinivasan e Thompson. Nas classes netgen-quarter e netgen-medium o desempenho permaneceu muito próximo ao desempenho nas classes com parametrização genérica. O desempenho da abordagem com ciclos de razão mínima melhorou um pouco ficando com os tempos de execução maiores por uma ordem de 30 em relação a abordagem de Srinivasan e Thompson (veja tabelas 6.4, 6.5 e 6.6).

Neste estudo experimental pode-se comprovar a influência da densidade do grafo na complexidade de resolução do PFCM paramétrico. Dentre todas as classes avaliadas as que se apresentaram serem as mais difíceis foram a classe torus $\sqrt[3]{n^5}$ da família torus e as classes da família netgen. As instâncias destas classes foram as que obtiveram os maiores tempos de execução devido ao maior número de fluxos ótimos computados. As instâncias com os menores tempos de execução pertenceram a família Grid-graph. Justamente a família onde todas as classes estão num formato de grade, um formato que implica em grafos mais esparsos. Esta esparsividade também implicou numa degeneração menor do que o observado nas demais classes. O pior resultado para a abordagem com ciclos de razão mínima foi para a família Grid-graph. Nesta família a diferença entre os tempos de execução entre as duas abordagens chegou a ser maior que 200 vezes.

De um modo geral se observou em todas instâncias que o número de fluxos ótimos computados permaneceu em função linear do tamanho das instâncias. A diferença na quantidade de fluxos ótimos para instâncias de mesmo tamanho em todas as classes avaliadas variou entre 20% e 50%. O maior número de fluxos ótimos computados foi 442 para uma instância da classe torus-$\sqrt[3]{n^5}$ com 80 vértices e parametrização genérica. Nas instâncias com parametrização 0-1 com exceção da família Grid-graph todas as demais classes apresentaram um número de fluxos computados no mínimo a metade do observado na respectiva classe com parametrização genérica. Para as instâncias com parametrização 0-1 o maior número de fluxos ótimos computados foi 118 também com uma instância da classe torus-$\sqrt[3]{n^5}$ com 80 vértices.

A degeneração nas instâncias de todas as classes mostrou ser independente do tipo de parametrização aplicada e dependente da topologia. A degeneração apesar de ser uma realidade na maioria das instâncias avaliadas não torna a abordagem de Srinivasan e Thompson uma solução inviável na prática. Em relação à abordagem com ciclos de razão mínima o seu comportamento mostrou estar relacionado somente ao número de

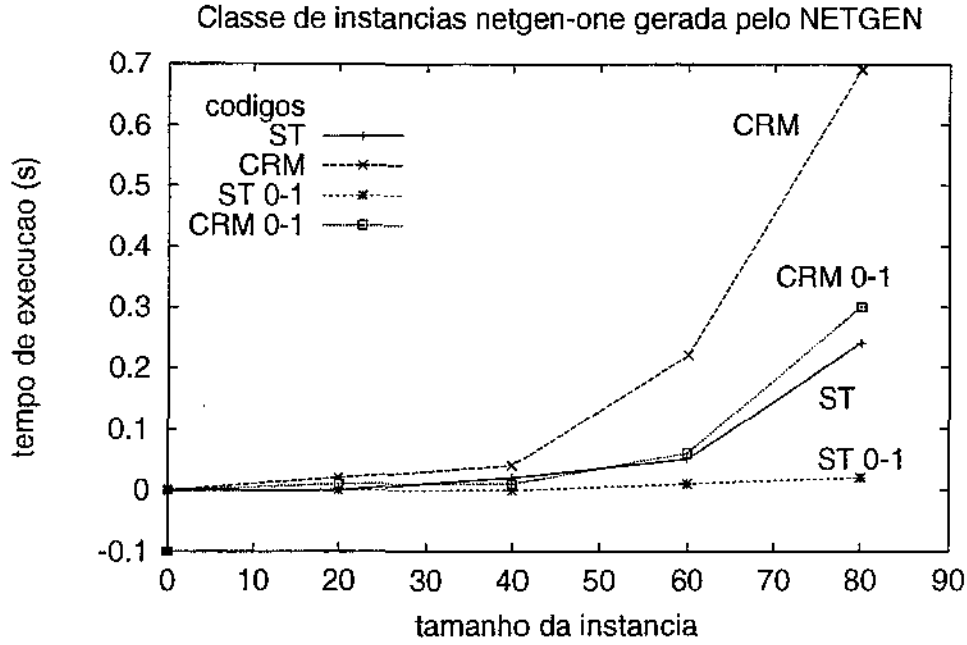


Figura 6.1: *Desempenho das abordagens na classe netgen-one*

fluxos ótimos computados para uma instância. Percebeu-se que a computação de ciclos de razão mínima é bastante onerosa. Na implementação realizada o tempo na prática para a computação de um ciclo de razão mínima foi sempre superior ao tempo da computação da primeira solução do problema que consistia na resolução de uma instância do PFCM. Um fator não esperado se considerarmos que a complexidade computacional do problema do ciclo de razão mínima é inferior à complexidade do problema do fluxo de custo mínimo. Este fato torna a abordagem com ciclos de razão mínima mais ineficaz na prática do que o método de Eisner e Severance.

Desconsiderando o aspecto do código utilizado para a abordagem de Srinivasan e Thompson ser excelente e que a implementação realizada para a abordagem com ciclos de razão mínima pode ser melhorada, a computação de pivotamentos é muito mais eficiente do que a computação de ciclos de razão mínima. Outro aspecto desfavorável em relação à abordagem com ciclos de razão mínima pode ser declarado. A conversão dos custos das arestas para números inteiros torna as iterações do algoritmo progressivamente mais lentas à medida que os custos vão se tornando maiores. Quanto maior for o número de fluxos ótimos computados pior tende a ser o desempenho da abordagem com ciclos de razão mínima.

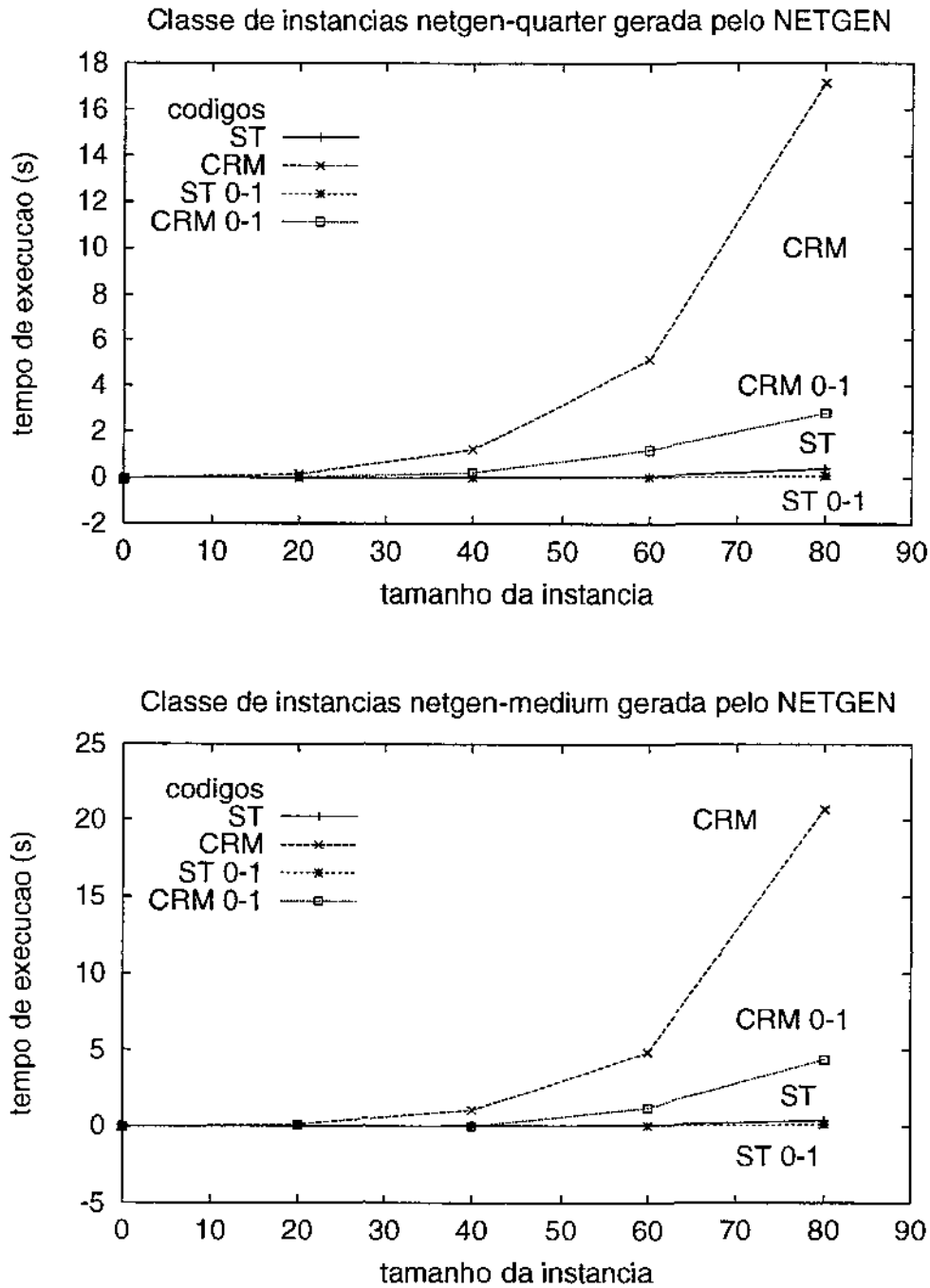


Figura 6.2: *Desempenho das abordagens nas classes netgen-quarter e netgen-medium*

Capítulo 7

Conclusão

Nesta dissertação se discutiram os aspectos a serem abordados para a resolução eficiente do problema da designação paramétrico. Estes aspectos incluem estabelecer a faixa de valores do parâmetro para o qual um emparelhamento permenece ótimo e estabelecer a partir de então os novos emparelhamentos ótimos. Infelizmente, não se conseguiu propor soluções eficientes para tratar todos os aspectos discutidos. A principal dificuldade encontrada foi computar as informações para a resolução do problema com uma complexidade baixa como $O(nm)$.

Uma nova abordagem ao problema da designação paramétrico foi proposta. Apesar de não implicar em resultados relevantes acreditamos que esta abordagem contribui em aumentar as diversas facetas que podem ser atacadas na realização de estudos futuros para o problema.

Uma importante relação entre o problema da designação paramétrico e o problema do ciclo de razão mínima foi demonstrada. Este relacionamento tem diversas implicações diretas. Primeiramente temos mais um relacionamento entre o problema do ciclo de razão mínima e problemas combinatórios paramétricos (o primeiro relacionamento foi demonstrado por Karp e Orlin [KO81]). Desenvolvimentos algorítmicos na complexidade para o PCRM poderão eventualmente implicar em novos resultados para PFCM ou PD paramétrico. Espera-se que mais esta aplicação do problema do ciclo de razão mínima seja um incentivo no surgimento de novos estudos sobre o problema.

A nova abordagem proposta para o problema do fluxo de custo mínimo paramétrico estendendo a abordagem para o problema da designação paramétrico demonstra que cada solução do problema pode ser computada em tempo inferior ao tempo de uma instância do problema padrão supondo que a hipótese de similaridade é satisfeita.

As implementações realizadas com o intuito de avaliar a sensibilidade das abordagens em relação às classes de instâncias paramétricas e às estruturas da rede concluíram que na prática a abordagem com ciclos de razão mínima é fraca. Os resultados obtidos em

instâncias de pequena magnitude nos levam a crer que resultados mais favoráveis não serão obtidos em instâncias maiores.

O fraco desempenho da abordagem com ciclos de razão mínima pode ser explicado de várias maneiras. A primeira delas é a robustez da abordagem de Srinivasan e Thompson. A eficiência na implementação dos pivotamentos e a degeneração observada nas classes avaliadas sugerem que instâncias de magnitude razoável poderão ser resolvidas eficientemente. Apesar do desempenho da abordagem de Srinivasan e Thompson variar com a estrutura das instâncias esta manteve-se robusta em todos os testes realizados.

Um segundo aspecto é a implementação realizada. Apesar de utilizar códigos altamente eficientes a implementação da abordagem com ciclos de razão mínima tem que realizar processamentos dispendiosos como a inicialização das estruturas de dados de cada código. A adequação dos códigos não é um argumento preponderante no fraco desempenho da abordagem com ciclos de razão mínima mas com certeza implementações mais eficientes podem ser obtidas com implementações específicas da abordagem.

Um terceiro aspecto é a detecção de ciclos negativos. A heurística utilizada foi capaz de reduzir o número total de iterações no procedimento de busca binária no algoritmo de Lawler. Entretanto, a detecção de ciclos negativos é ainda um processamento oneroso na abordagem com ciclos de razão mínima. A obtenção de uma implementação eficiente desta abordagem dependerá da computação eficiente de ciclos negativos, seja através de novas heurísticas na redução de iterações no algoritmo de Lawler ou na computação de ciclos negativos.

Por último, espera-se que se obtenham algoritmos mais eficientes para a computação de ciclos de razão mínima tanto no ponto de vista teórico quanto do ponto de vista prático. A diferença entre os tempos dos algoritmos de tempo polinomial forte para o problema do ciclo de razão mínima e o seu caso específico o problema do ciclo de média mínima é da ordem de $O(n \log n)$.

Apêndice A

Geradores

A.1 Grid-on-torus

O **Grid-on-torus** desenvolvido por A. V. Goldberg [GK93] produz instâncias do PFCM capacitadas. Este gerador foi desenvolvido com a finalidade de produzir instâncias de difícil resolução do PFCM. As instâncias obtidas, aparentemente, estabelecem dificuldades para os algoritmos combinatórios do problema.

A estrutura básica para a instância é uma grade retangular definida em função do número de vértices. A altura da grade é estabelecida aproximadamente como $h \approx \sqrt[3]{n^2}$ e a largura como $w \approx \sqrt[3]{n}$ de maneira que tenhamos a seguinte relação $hw < n$.

O vértice fonte é conectado aos vértices da primeira coluna da grade retangular e o sorvedouro conectado aos vértices da última coluna. Os custos e as capacidades das arestas são escolhidos randomicamente de distribuições uniformes que dependem dos parâmetros de entrada.

Os parâmetros de entrada foram reduzidos e restrições foram adicionadas para facilitar a utilização do gerador. Os parâmetros de entrada para o *Grid-on-torus* são os seguintes:

$n \geq 15$	número de vértices
$6n \leq m \leq \sqrt[3]{n^5}$	número de arestas
$U \geq 8$	capacidade máxima de uma aresta
$C \geq 8$	custo máximo de uma aresta
s	semente para o gerador de números randômicos

Cada vértice é conectado a todos os demais na mesma linha (arestas horizontais) idades geralmente altas escolhidas no intervalo $[1, U]$.

O excesso de fluxo na fonte é aproximadamente igual a capacidade do corte da última coluna. A factibilidade da instância é assegurada pela existência de um caminho hamiltoniano[1] da fonte para o sorvedouro com arestas de capacidades e custos altos. As instâncias

[1] Caminho contendo todos os vértices do grafo

geradas pelo *Grid-on-torus* se caracterizam pelo fato de que para qualquer caminho, existem usualmente muitos caminhos cujos custos e capacidades são próximos, similarmente para os cortes entre a fonte e o sorvedouro.

A.2 Grid-graph

O **Grid-graph** produz instâncias do PFCM no formato de grade com uma fonte e um sorvedouro. O *Grid-graph* escrito em FORTRAN por Resende e Veiga [RV93] é baseado no gerador proposto por Karmakar e Ramakrishnan [KR91]. Algumas subrotinas utilizadas na sua implementação estão descritas nos trabalhos de Goldfarb e Grigoriadis [GG88][2] e Schrage [Sch79][3].

O *Grid-graph* gera uma instância com $hw + 2$ vértices. O formato da instância é uma grade retangular com dimensões hxw mais uma fonte e um sorvedouro.

Os parâmetros de entrada do *Grid-graph* são:

h	altura da grade
w	largura da grade
C	custo máximo de uma aresta
U	capacidade máxima de uma aresta
s	semente para o gerador de números randômicos

O vértice fonte é conectado aos vértices da primeira coluna por arestas orientadas não capacitadas de custo nulo. Similarmente, tem-se arestas da última coluna para o sorvedouro. O excesso de fluxo na fonte e a demanda no sorvedouro são definidas como sendo igual ao fluxo máximo no grafo entre estes vértices.

Na grade retangular todos os vértices são de entreposto. Cada vértice é conectado aos vértices vizinhos na grade por arestas orientadas da direita para a esquerda e de cima para baixo. Os custos das arestas são escolhidas uniformemente no intervalo $[0, C]$ e as capacidades no intervalo $[0, U]$.

A.3 Netgen

O clássico gerador descrito por Klingman, Napier e Stutz [KNS74] denominado **Netgen** é um dos geradores mais utilizados para a construção de instâncias para os problemas de Fluxo em Redes. A versão em C do *Netgen* utilizada neste estudo foi implementada por N. Schlenker.

[2]Rotina para o gerador de números randômicos

[3]Rotina para computar o fluxo máximo no grafo

O *Netgen* produz instâncias capacitadas e não capacitadas do PFCM, do PT, dentre outros. Este gerador possibilita gerar classes de instâncias estruturalmente distintas com diversas características específicas.

O processo de geração de instâncias do *Netgen* inicia-se primeiramente com a obtenção de um grafo de acordo com a estrutura definida pelos parâmetros de entrada. Este grafo contém o número estabelecido para cada tipo de vértice e o excesso total de fluxo. O grafo obtido é gerado de maneira a ser o mais esparso possível e a garantir que a instância resultante seja conectada e factível.

Os parâmetros de entrada para o *Netgen* relativas ao tamanho da instância são:

n	número de vértices
n_s	número de vértices fontes puros
n_t	número de vértices sorvedouros puros
n_{ts}	número de fontes de entreposto
n_{tt}	número de sorvedouros de entreposto
m	número de arestas

O grafo inicial servirá como um esqueleto para a instância a ser gerada, restando definir posteriormente as características da instância de acordo com os demais valores estabelecidos para os parâmetros de entrada. Todos as arestas no esqueleto são capacitadas de maneira que o montante de fluxo especificado pode ser enviado usando somente estas arestas. Esta característica permite a existência de soluções factíveis para o problema com uma pequena porcentagem de arestas saturadas. Por último, a instância é circularizada adicionando-se uma super-fonte e um super-sorvedouro.

Os parâmetros de entrada contendo as informações relativas a estrutura do problema são:

b	excesso total
$\underline{c}$	custo mínimo de uma aresta
$\overline{c}$	custo máximo de uma aresta
$\underline{u}$	capacidade mínima de uma arestas
$\overline{u}$	capacidade máxima de uma arestas
$\%_c$	porcentagem das arestas do esqueleto com custo máximo
$\%_u$	porcentagem das arestas capacitadas
s	semente para o gerador de números randômicos

A.4 GRCP

O **Grcp** parametriza instâncias de problemas de Fluxo em Redes no formato padrão do *DIMACS*. O gerador tem como entrada um arquivo no formato padrão do *DIMACS* e alguns parâmetros referentes as características da parametrização a ser realizada. Como

saída o *Grcp* cria um novo arquivo com um campo a mais referente a variação do custo em relação ao parâmetro.

O gerador busca desenvolver instâncias do PFCM paramétrico de difícil resolução. A influência do parâmetro sobre o custo das arestas é definida de maneira a evitar que o número de fluxos ótimos distintos obtidos na resolução do problema seja reduzido.

Os argumentos de entrada do *Grcp* relacionados a parametrização dos custos tratam-se dos seguintes:

bc	parâmetro para a função de parametrização das arestas de baixo custo
ac	parâmetro para a função de parametrização das arestas de alto custo
C	maior valor permitido para a parametrização dos custos
s	semente para o gerador de números randômicos

A metodologia aplicada até o momento para criar dificuldade nas instâncias é permitir uma influência maior do parâmetro nas arestas de baixo custo e vice-versa. Evitamos assim que uma parcela das arestas do problema seja irrelevante na resolução do problema. Isto ocorreria se uma aresta mantivesse o seu custo muito baixo ou muito alto para todo o intervalo do parâmetro. Quanto maior for o número de arestas candidatas a estarem num fluxo ótimo maior será a possibilidade de encontrarmos mais fluxos distintos para o intervalo em questão.

Este gerador classifica as arestas como de baixo custo e alto custo com base na faixa de valores atribuídos como custos das arestas $[\underline{C}, \overline{C}]$. As arestas com custos inferiores a $\frac{\underline{C}+\overline{C}}{2}$ tem os coeficientes do parâmetro definidos pela função

$$R(s)^{bc}\, C$$

onde $R(s)$ é um gerador de números randômicos. Para as arestas com custo superior ou igual a $\frac{\underline{C}+\overline{C}}{2}$ temos que os coeficientes são definidos pela função

$$\sqrt[ac]{R(s)}\, C$$

Para os testes realizados utilizou-se $R(s)^2$ para as arestas de baixo custo e $\sqrt{R(s)}$ para as arestas de alto custo.

Índice

Bibliografia

[AGOT92] R. K. Ahuja, A. V. Goldberg, J. B. Orlin, and R. E. Tarjan. Finding minimum-cost flows by double scaling. *Mathematical Programming*, 53(3):243–266, 1992.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows - Theory, Algorithms and Applications.* Prentice Hall Inc, New Jersey, 1993.

[AMOT90] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2):213–223, 1990.

[Bal95] V. K. Balakrishnan. *Network Optimization.* Chapman & Hall Mathematics Series, 1995.

[Bel58] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[Ber57] C. Berge. Two theorems in graph theory. In *Proceedings of the National Academic of Sciences*, volume 43, pages 842–844, USA, 1957.

[Ber79] D. P. Bertsekas. A distributed algorithm for the assignment problem. *Working Paper Laboratory for Information and Decision Systems MIT*, 1979.

[Ber91] D. P. Bertsekas. *Linear Network Optimization.* Prentice-Hall Inc, New Jersey, 1991.

[BGAB83] L. D. Bodin, B. L. Golden, A. A. Assad, and M. O. Ball. Routing and scheduling of vehicles and crews. *Computers and Operations Research*, 10:65–211, 1983.

[BT76] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear diophantine equations. In *Proceedings of American Mathematical Society*, volume 55, pages 299–304, 1976.

[BT89] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation - Numerical Methods.* Prentice-Hall Inc, New Jersey, 1989.

[CLR90] T. H. Cormen, C. L. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT-Press, New York, 1990.

[Dan51] G. B. Dantzig. Application of the simplex method to a transportation problem. In T. C. Koopmans, editor, *Activity analysis and production and allocation*, pages 359–373. Wiley, New York, 1951.

[DBR67] G. B. Dantzig, W. Blattner, and M. R. Rao. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. In P. Rosenstiehl, editor, *Theory of graphs*, pages 77–84. Dunod, Paris and Gordon and Breach, New York, 1967. International Symposium.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[Egé31] J. Egérváry. Matrixok kombinatórius tulajdon. *Mat. Fiz. Lapok*, 38:16–28, 1931.

[EK72] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[ES76] M. J. Eisner and D. G. Severance. Mathematical techniques for efficient record segmentation in large shared databases. *Journal of the ACM*, 23(4):619–635, 1976.

[FF57] L. R. Ford and D. R. Fulkerson. A primal-dual algorithm for the capacitated hitchcock problem. *Naval Reserach Logistics Quarterly*, 4:47–54, 1957.

[FF62] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, New Jersey, 1962.

[FT87] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[GG88] D. Goldfarb and M. D. Grigoriadis. A computational comparison of the dinic and network simplex methods for maximum flow. In *Annals of Operations Research*, volume 7, pages 83–123, 1988.

[GGT89] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A faster parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[GJ79] M. S. Garey and D. S. Johnson. *Computers and Intractibility - A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[GK93] A. V. Goldberg and M. Kharitanov. On implementing scaling push-relabel algorithms for the minimum-cost flow problems. In D. S. Johnson and C. C. McGeoch, editors, *Network flows and Matching*, volume 12, pages 157–198. American Mathematical Society, 1993.

[GM84] M. Gondran and M. Minoux. *Graphs and Algorithms.* Wiley - Interscience Series in Discrete Mathematics and Optimization, New York, 1984.

[Gol97] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.

[GPV93] A. V. Goldberg, S. A. Plotkin, and P. M. Vaidya. Sublinear-time parallel algorithms for matching and related problems. *Journal of Algorithms*, 14(2):180–213, 1993.

[GR98] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of ACM*, 45(6):755–782, 1998.

[GT88] A. V. Goldberg and R. T. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35(4):921–940, 1988.

[GT89a] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

[GT89b] A. V. Goldberg and R. T. Tarjan. Finding minimum cost circulations by cancelling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.

[GT90] A. V. Goldberg and R. E. Tarjan. Solving minimum cost flow problem by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[Gus83] D. Gusfield. Parametric combinatorial computing and a problem of a program module distribution. *Journal of the ACM*, 30(3):551–563, 1983.

[Joh82] D. B. Johnson. A priority queue in which initialization and queue operations take $O(\log \log D)$. *Mathematical Systems Theory*, 15(4):295–309, 1982.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thacher, editors, *Complexity of Computer Computations*, pages 83–103. Plenum Press, New York, 1972.

[Kar78] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978.

[KH80] J. L Kennington and R. V. Helgason. *Algorithms for Network Programming.* John Wiley and Sons, 1980.

[Kle67] M. Klein. A primal method for minimal cost flow with application to the assignment and transportation problems. *Management Science*, 14:205–220, 1967.

[KNS74] D. Klingman, A. Napier, and J. Stutz. Netgen: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.

[KO81] R. M. Karp and J. B. Orlin. Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Applied Mathematics*, 3(1):37–45, 1981.

[Kön31] D. König. Graphen und matrizen. *Mat. Fiz. Lapok*, 38:116–119, 1931.

[Kön50] D. König. *Theorie der Endlichen und Unendlichen Graphen.* Chelsea, 1950.

[KR91] N. K. Karmakar and K. G. Ramakrishnam. Computational results of an interior point algorithm for large scale linear programming. *Mathematical Programming*, 52(3):555–586, 1991.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[Kuh56] H. W. Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3:253–258, 1956.

[Law67] E. L. Lawler. Optimal cycles in doubly weighted linear graphs. In P. Rosenstiehl, editor, *Theory of graphs*, pages 209–214. Dunod, Paris and Gordon and Breach, New York, 1967. International Symposium.

[Law71] E. L. Lawler. A solvable case of the traveling salesman problem. *Mathematical Programming*, 1(2):267–269, 1971.

[Law73] E. L. Lawler. Optimal cycles in graphs and the minimal cost-to-time ratio problem. In *Proceedings of a Conference on Periodic Optimization*, Udine, Italy, 1973. CISM.

[Law76] E. L. Lawler. *Combinatorial Optimization: Network and Matroids.* Holt, Rinehart, and Winston, New York, 1976.

[LLKS85] E. L . Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley - Interscience Series in Discrete Mathematics and Optimization, New York, 1985.

[LN87] G. Laporte and Y. Nobert. Exact algorithms for the vehicle routing problem. In S. Martello, G. Laporte, M. Minoux, and C. Ribeiro, editors, *Surveys in Combinatorial Optimization.* North-Holland, 1987.

[Man89] U. Manber. *Introduction to Algorithms - An Creative Approach.* Addison-Wesley Publishing Company Inc, Massachusetts, 1989.

[Meg79] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979.

[MP89] D. L. Miller and J. F. Pekny. Results from a parallel branch and bound algorithm for solving large asymmetric traveling salesman problems. *Operations Research Letters*, 8(3):129–135, 1989.

[NR59] R. Z. Norman and M. O. Rabin. An algorithm for a minimum cover of a graph. *Proccedings of American Mathematical Society*, 10:315–319, 1959.

[NW89] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley - Interscience Series in Discrete Mathematics and Optimization, New York, 1989.

[OA92] J. B. Orlin and R. K. Ahuja. New scaling algorithms for the assignment and minimum cycle mean problems. *Discrete Applied Mathematics*, 54(1):41–56, 1992.

[Orl88] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 337–387, 1988.

[Orl97] J. B. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, 1997.

[PRW93] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment Problem and Related Problems*, volume 17, pages 1–42. American Mathematical Society, 1993.

[PS98] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity.* Dover Publications, New Jersey, 1998.

[RV93] M. G. C. Resende and G. Veiga. An efficient implementation of a network interior point method. In D. S. Johnson and C. C. McGeoch, editors, *Network flows and Matching*, volume 12, pages 299–348. American Mathematical Society, 1993.

[Sch79] L. Schrage. A more portable fortran random number generator. *ACM Transactions on Mathematical Software*, June, 1979.

[SS89] H. Schneider and M. H. Schneider. Max-balacing weighted directed graphs. Unpublished manuscript, University of Wisconsin, 1989.

[ST72] V. Srinivasan and G. L. Thompson. An operator theory of parametric programming for the transportation problem. *Naval Research Logistics Quarterly*, 19:205–252, 1972.

[Tar83] R. E. Tarjan. *Data Structures and Network Algorithms.* Soc. Indust. Appl. Math, Philadelphia, 1983.

[Tar85] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[Tom72] N. Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1:173–194, 1972.

[Wes96] D. B. West. *Introduction to Graph Theory.* Prentice-Hall Inc, New Jersey, 1996.

[WZ91] C. Wallacher and U. T. Zimmerman. A combinatorial interior point method for network flow problems. In *14th International Symposium on Mathematical Programming*, volume 21, pages 37–45, Amsterdam, 1991. The Netherlands.

[YTO91] N.E. Young, R. E. Tarjan, and J.B. Orlin. Faster parametric shortest-path and minimum-balance algorithms. *Networks*, 21(2):205–221, 1991.