Impacto da reordenação de matrizes por nested dissection nos métodos de pontos interiores

Wellington Barbosa Rodrigues

Dezembro / 2016

Dissertação de Mestrado em Ciência da Computação

Impacto da reordenação de matrizes por *nested dissection* nos métodos de pontos interiores

Este documento corresponde à dissertação apresentada à Banca Examinadora no curso de Mestrado em Ciência da Computação da Faculdade Campo Limpo Paulista.

Campo Limpo Paulista, 30 de dezembro de 2016.

Wellington Barbosa Rodrigues

Profa. Dra. Marta Ines Velazco Fontova
(Orientadora)

Faculdade Campo Limpo Paulista
Programa de Mestrado em Ciência da Computação

"Impacto da reordenação de matrizes por nested dissection nos métodos de pontos interiores"

Wellington Barbosa Rodrigues

Dissertação de Mestrado apresentado ao Programa de Mestrado em Ciência da Computação da Faculdade Campo Limpo Paulista, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

_____

Profa. Dra. Marta Ines Velazco Fontova
(Orientadora - FACCAMP)

_____

Profa. Dra. Maria do Carmo Nicoletti
(FACCAMP)

_____

Prof. Dr. Aurelio Ribeiro Leite de Oliveira
(IMECC-UNICAMP)

Campo Limpo Paulista, 30 de dezembro de 2016.

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP)

Câmara Brasileira do Livro, São Paulo, Brasil.

Rodrigues, Wellington Barbosa
Impacto da reordenação de matrizes por nested dissection nos métodos de pontos interiores / Wellington Barbosa Rodrigues. Campo Limpo Paulista, SP: FACCAMP, 2016.

Orientadora: Profª. Drª. Marta Ines Velazco Fontova
Dissertação (Programa de Mestrado em Ciência da Computação) – Faculdade Campo Limpo Paulista – FACCAMP.

1. Pontos interiores. 2. Nested dissection. 3. Reordenação. I. Velazco Fontova, Marta Inez. II. Campo Limpo Paulista. III. Título.

CDD-005.1

Resumo: *Os métodos de pontos interiores são eficientes na solução de problemas de programação linear de grande porte. O cálculo das direções de busca requer a solução de um ou mais sistemas lineares. Este é o passo mais caro computacionalmente; os sistemas envolvem matrizes esparsas e mal condicionadas. Para solução dos sistemas lineares podem ser utilizados métodos diretos por fatoração de Cholesky ou métodos iterativos por meio do método dos gradientes conjugados precondicionado. Resultados mostram que a reordenação da matriz do sistema traz vantagens no cálculo da solução por métodos diretos. Este estudo investiga o impacto do método de ordenação nested dissection, através da comparação com o método mínimo grau, na solução de sistemas lineares oriundos de métodos de pontos interiores pelo método iterativo dos gradientes conjugados precondicionado.*

Palavras-chave: *Pontos interiores; Nested Dissection; Reordenação.*
Linha de Pesquisa: *Pesquisa operacional.*

Abstract: *Interior point methods are efficient for solving large-scale linear programming problems. The computation of search directions requires the solution of one or more linear systems. This is the most computationally expensive step; the systems involve sparse and ill conditioned matrices. The linear systems can be solved by direct methods with Cholesky decomposition or iterative methods with conjugate gradient preconditioned method. Preview results show that matrix reordering brings advantages in the solution by direct methods. This study investigates the impact of nested dissection by comparison with a minimum ordering, in the solution of linear systems, arising from interior point methods using the preconditioned conjugate gradient method.*

Keyword: *Interior Point methods, Nested Dissection, Reordering.*
Line of Research: *Operational Research.*

Agradecimentos

Agradeço primeiramente a minha mãe, irmãos e sobrinhos pelo apoio durante todos os períodos da minha vida e pela compreensão em meus momentos de ausência e estresse durante o mestrado.

À minha professora e orientadora Marta Ines Velazco Fontova, por sua dedicação, compreensão e conhecimentos transmitidos durante todo o processo de orientação, servindo como exemplo acadêmico, profissional e pessoal que irei levar para o resto da vida.

Ao Paulo Matsushita, meu mentor na área de computação, responsável pelos meus primeiros passos profissionais nessa área e responsável por diversos incentivos que me levaram à produção deste trabalho.

Ao Rafael Santos e Edilania Medeiros, por seu apoio e incentivo no processo de entrada no programa de mestrado.

A todos os professores e colegas que estiveram comigo durante esse processo, em especial Leonardo Cortez, por seus conhecimentos matemáticos e conselhos pessoais. Ao Sérgio Santos pela sua disposição em ajudar a todos, sempre. Ao André Barone pelo seu conhecimento e disposição em me ajudar durante a fase de implementação. Ao Rodrigo Ramos e Ricardo Araújo que foram companheiros inestimáveis durante as disciplinas e ao João Cruz, por todos os dias de estudos, conversas e apoio durante a escrita dessa dissertação.

Sumário

Glossário

FCC – Fatoração Controlada de Cholesky.

GCP – Gradiente Conjugado Precondicionado.

MG – Mínimo Grau.

MPI – Métodos de Pontos Interiores

ND – Nested Dissection.

PCx-MG – PCx Modificado com reordenação por Mínimo Grau.

PCx-ND – PCx Modificado com reordenação por Nested Dissection.

Lista de Tabelas

Lista de Figuras

1. INTRODUÇÃO

Um problema de programação linear busca a minimização ou maximização de uma função linear respeitando um conjunto de restrições lineares que formam a região factível do problema (Bazaraa, et al., 2010).

Problemas de programação linear são muito comuns em diversas áreas da ciência, como computação, engenharias e economia. Com os avanços tecnológicos e a evolução da sociedade, os problemas estão se tornando cada vez mais complexos e, consequentemente, envolvendo um maior custo computacional. Um exemplo é o problema de alocação da tripulação em aviões (Gomes, 2009). Este tipo de problema tem diversas restrições que precisam ser levadas em consideração no momento da modelagem, como jornada máxima de trabalho permitida, folga da tripulação, quantidade máxima de horas de voo e afins.

Em 1947, Dantzig definiu o método simplex para a solução de problemas de programação linear. Por muitos anos, este foi o único método disponível para solução deste tipo de problema (Bazaraa, et al., 2010). O As método simplex a cada iteração calcula uma solução básica factível ou ponto extremo do polítipo convexo definido pelas restrições do problema, até a solução ótima ou solução ilimitada. No pior caso, ele percorre todos os pontos extremos, o que o caracteriza como um método de ordem não polinomial (Vanderbei, 2014).

No ano de 1979, Khachiyan (Khachiyan, 1980) desenvolveu o método dos elipsoides de ordem polinomial para solução de problemas de programação linear. O método teve grande impacto na área de programação linear, mas a sua implementação não era competitiva com o método simplex devido ao seu alto custo algébrico (Ross, et al., 2005).

Em 1984, Karmarkar (Karmarkar, 1984) publicou um novo método, de ordem polinomial, para a solução deste tipo de problema. Este trabalho abriu uma nova área de pesquisa e, no período de 1984 a 1989, foram publicados mais de 1.300 artigos sobre o assunto (Ross, et al., 2005). Estes métodos, como o próprio nome diz, se limitam exclusivamente aos pontos interiores da região factível do problema.

O método primal-dual, em sua variação preditora-corretora (Mehrotra, 1992), exige a solução de dois sistemas lineares a cada iteração, para o cálculo das direções de busca, e é seu passo de maior custo computacional. Os métodos de pontos interiores primais-duais possuem melhores propriedades teóricas que facilitam as análises de complexidade e convergência e são os mais eficientes para a solução de problemas práticos (Vanderbei, 2014).

Os sistemas lineares oriundos de métodos de pontos interiores para o cálculo das direções podem ser resolvidos por métodos diretos ou métodos iterativos. Os métodos diretos encontram a solução do sistema em um número predeterminado de passos. Na solução por métodos iterativos, a cada iteração é obtida uma nova solução que forma uma sequência convergente à solução do sistema. O número de iterações não é previamente conhecido.

O passo mais caro nos métodos primais-duais é a solução destes sistemas lineares.

Em problemas esparsos e de grande porte, quando os sistemas lineares são resolvidos por métodos diretos podem acontecer problemas de uso excessivo de memória para armazenamento e processamento. Uma forma de amenizar este tipo de problema é por meio do uso de métodos de reordenação da matriz do sistema.

Na solução dos sistemas por métodos iterativos, reordenar a matriz dos sistemas pode ajudar na convergência do método.

A Figura 1.1 mostra a relação entre o método preditor-corretor e os métodos de solução de sistemas lineares. No método preditor-corretor, as direções de busca são calculadas através da solução de dois sistemas lineares. Estes sistemas podem ser resolvidos por métodos diretos ou métodos iterativos.

1.1. Métodos de reordenação e métodos de pontos interiores

Os sistemas lineares oriundos de métodos de pontos interiores envolvem a mesma matriz dos coeficientes. Devido à formulação desses sistemas, a matriz dos coeficientes é muito esparsa e o padrão de esparsidade é mantido durante todas as iterações dos métodos de pontos interiores. Por outro lado, tanto o valor dos elementos da matriz quanto a diferença entre eles variam a cada iteração. Especificamente, a

diferença relativa entre os elementos da matriz pode ser muito grande nas últimas iterações dos métodos de pontos interiores. Esse fato caracteriza a matriz como mal

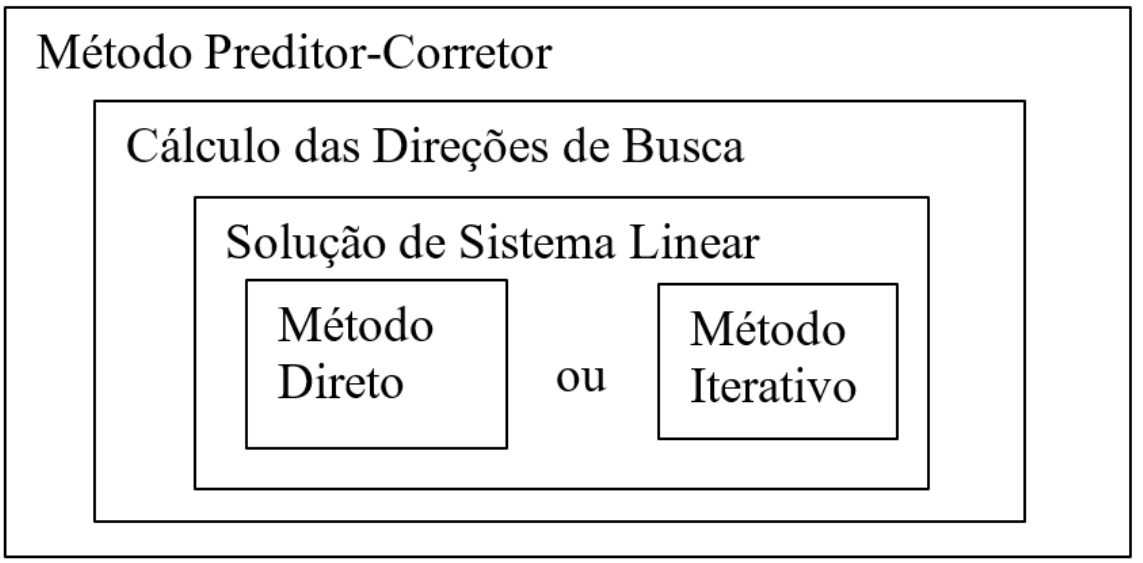

Figura 1.1 – Relação entre métodos de solução de sistemas lineares e o método preditor-corretor.

condicionada.

Adicionalmente, esta matriz é simétrica e definida positiva, o que permite o uso de métodos específicos para a solução de tais sistemas lineares.

Para a solução dos sistemas por métodos diretos é utilizada a fatoração de Cholesky. Esta fatoração, em problemas esparsos cria fatores com maior quantidade de elementos não nulos do que a matriz original (George, et al., 1994). Quanto maior o preenchimento do fator, maior será a necessidade de recursos computacionais para o seu armazenamento e processamento.

Visando reduzir o preenchimento na solução de sistemas lineares com matrizes esparsas são utilizados métodos de reordenação (Davis, et al., 2016). Suponha uma matriz quadrada e simétrica de dimensão n com uma ordenação inicial representada pelos índices de 1 até n de suas linhas/colunas. Um método de reordenação calcula uma nova sequência de índices das linhas/colunas mantendo a simetria da matriz. Existem diversos métodos para reordenação de matrizes, como *nested dissection* (ND) (George, 1973), mínimo grau (MG) (Tinney & Walker, 1967), *Reverse Cuthill-McKee* (Cuthill & McKee, 1969) e *Sloan* (Sloan, 1986).

Para matrizes oriundas de métodos de pontos interiores, em (Rothberg & Hendrickson, 1998) foi utilizada uma reordenação híbrida entre ND e MG que resultou em uma redução na quantidade de cálculos para a solução dos sistemas por métodos diretos.

Na solução dos sistemas de pontos interiores por métodos iterativos é, geralmente, utilizado o método dos gradientes conjugados (Saad, 2003). Este método possui uma convergência muito rápida para a solução do sistema quando a matriz envolvida é bem condicionada. O condicionamento da matriz dos métodos de pontos interiores varia muito no processo iterativo e, nas ultimas iterações, torna-se muito mal condicionada, o que leva à necessidade do precondicionamento da mesma.

Bocanegra desenvolveu um precondicionador específico para a matriz dos métodos de pontos interiores (Bocanegra, 2005). O precondicionador híbrido desenvolvido por ela, com as variações propostas por (Velazco et al., 2010), mostraram resultados satisfatórios na solução de problemas de grande porte de programação linear.

A reordenação da matriz dos coeficientes pode melhorar a qualidade dos precondicionadores baseados em fatorações incompletas (neste caso, o precondicionador é obtido a partir de uma fatoração quase completa da matriz do sistema). Carmo (Carmo, 2005) avaliou a influência de três métodos de reordenação na resolução de sistemas lineares com o método iterativo Cholesky controlado gradiente conjugado: contagem de coluna, *Cuthill-McKee* (Cuthill & McKee, 1969) e MG. O método Cholesky controlado gradiente conjugado convergiu com um menor número de iterações na solução dos sistemas reordenados com o MG.

Em (Silva, 2015) foram avaliados três métodos de reordenação na solução dos sistemas oriundos de métodos de pontos interiores pelo método gradiente conjugado precondicionado (GCP) com o precondicionador híbrido (Bocanegra, 2005). Os métodos de reordenação avaliados foram: *Reverse Cuthill-McKee*, MG e *Sloan*. Os resultados permitiram aos autores concluir que a utilização de métodos de reordenação, durante a fase de preprocessamento, gera redução na densidade da matriz após a fatoração incompleta de Cholesky, redução na quantidade total de iterações e redução no tempo de solução do problema. O método de reordenação com melhor desempenho foi o MG.

Porém, em (George, et al., 1994) é mostrado que o método de reordenação ND é superior ao MG em velocidade e redução do preenchimento. Adicionalmente, em (Lugon, 2013), foram utilizados os métodos de reordenação: *Sloan*, *reverse Cuthill-Mckee*, *espectral*, *ND* e *approximate minimum degree* para a solução dos sistemas com o método iterativo dos resíduos mínimos generalizados (GMRES) (Saad, 2003) com um precondicionador baseado na fatoração LU incompleta. Os melhores resultados foram obtidos com o ND.

Os resultados obtidos em estudos anteriores (Rothberg & Hendrickson, 1998), (Silva, 2015) mostram as vantagens do uso de métodos de reordenação na solução dos sistemas próprios de pontos interiores por métodos diretos e por métodos iterativos. Silva (Silva, 2015) concluiu que o método mais apropriado é o MG, porém o ND não foi incluído no estudo.

Este trabalho investigou o impacto do método de reordenação ND na solução dos sistemas de métodos de pontos interiores pelo GCP.

A investigação foi realizada por meio da comparação do desempenho dos métodos, ND e MG. Os métodos foram introduzidos na implementação PCx-Modificado (Czyzyk, et al., 1999), (Bocanegra, 2005), (Velazco et al., 2010). Esta implementação de programação linear é feita em linguagem C e Fortran do método preditor-corretor de pontos interiores onde os sistemas são resolvidos pelo GCP com o precondicionador híbrido.

Este documento está organizado como segue. No Capítulo 2 são apresentados os métodos de pontos interiores primais duais. Os sistemas lineares e seus métodos de solução são discutidos no Capítulo 3. O Capítulo 4 mostra os métodos de reordenação para matrizes esparsas que foram objeto desta pesquisa. Os resultados do estudo comparativo entre os métodos de reordenação em problemas de médio e grande porte são descritos no Capítulo 5. Por fim, no Capítulo 6 são apresentadas as conclusões e trabalhos futuros.

2. MÉTODOS DE PONTOS INTERIORES

2.1. Programação linear

Um problema de programação linear busca a otimização de uma função linear sujeita a restrições lineares. As restrições delimitam a região factível do problema. O problema pode ser representado na forma padrão como descrito pela Equação (2.1),

$$\begin{array}{ll} min & c^t x \\ sujeito\ a & Ax = b, \\ & x \geq 0 \end{array} \tag{2.1}$$

em que $A \in \mathbb{R}^{m \times n}$ é matriz de restrições de m linhas e n colunas, $c \in \mathbb{R}^n$ é o vetor de coeficiente da função objetivo, $b \in \mathbb{R}^m$ é o vetor das restrições e $x \in \mathbb{R}^n$ é o vetor de variáveis de decisão (Vanderbei, 2014). No contexto deste trabalho, será assumido que a matriz *A* tem posto completo (linhas não redundantes).

Associado ao problema descrito na Equação (2.1), pode ser definido um novo problema utilizando os mesmos dados A, b, c e com novas variáveis.

$$\begin{array}{ll} max & b^t y \\ sujeito\ a & A^t y + z = c, \\ & z \geq 0 \end{array} \tag{2.2}$$

em que $y \in \mathbb{R}^m$ é o vetor das variáveis duais livres e $z \in \mathbb{R}^n$ é o vetor das variáveis de folga. O problema representado pela Equação (2.1) é chamado de problema primal e o novo problema definido pela Equação (2.2) será o problema dual.

2.2. Condições de otimalidade

As condições de Karush-Kunch-Tucker (KKT) em programação linear, também conhecidas como condições de otimalidade, garantem as condições necessárias e suficientes para que um ponto (x^*, z^*, y^*) seja uma solução ótima dos problemas primal e dual (Vanderbei, 2014).

Considere os problemas primal (2.1) e dual (2.2). As condições de KKT, que garantem que os pontos (x^*, z^*, y^*) são soluções ótimas do primal e do dual, são definidas conforme a seguir:

1. $Ax^* = b$ $\quad x^* \geq 0$ Factibilidade do primal;
2. $A^t y^* + z^* = c$ $\quad z^* \geq 0$ Factibilidade do dual;
3. $X^* Z^* e = 0$ Complementaridade,

em que X e Z são matrizes diagonais. A diagonal da matriz X será o vetor x ($X = diag(x)$) e a diagonal de Z, o vetor z ($Z = diag(z)$). O vetor $e \in \mathbb{R}^n$ é o vetor unitário de dimensão *n*.

A primeira condição estabelece a factibilidade da solução primal e a segunda condição, a factibilidade da solução dual. Da condição de complementaridade pode ser deduzido que como as variáveis x^* e z^* são não negativas ($x^* \geq 0$ e $z^* \geq 0$) então a condição $X^* Z^* e = 0$ será satisfeita somente se $x_j^* z_j^* = 0,\ \forall j = 1, \dots, n$. Isto permite concluir que:

$$\text{Se } x_j^* > 0 \ \rightarrow \ z_j^* = 0,$$

$$\text{Se } z_j^* > 0 \ \rightarrow \ x_j^* = 0.$$

2.3. Métodos de pontos interiores

Em 1979 Khachiyan, (Khachiyan, 1980) propôs o primeiro algoritmo polinomial para programação linear; o método dos elipsóides. Esse método consiste na construção de elipsóides dentro da região factível. Os centros desses elipsóides geram uma sequência de pontos que convergem para uma solução ótima do problema. Embora esse método tenha causado grande impacto na área de programação linear não foi aceito como alternativa ao método simplex, pois o custo algébrico, por iteração, é muito alto (Gondzio, 2012).

Karmarkar em 1984 (Karmarkar, 1984) propôs o primeiro método de pontos interiores de ordem polinomial. Essa proposta foi o ponto de partida para novos métodos de pontos interiores.

Os métodos de pontos interiores trabalham exclusivamente nos pontos interiores ao ortante positivo; considerando problemas na forma padrão do primal e do dual serão tais que $x > 0$ e $z > 0$.

2.3.1. Métodos primais-duais

Os métodos primais-duais resolvem simultaneamente o problema primal e o problema dual.

Considere o sistema não linear formado pelas condições de otimalidade, desconsiderando as restrições de não negatividade como representado pela Equação (2.3).

$$F(x, y, z) = \begin{bmatrix} Ax - b \\ A^t + z - c \\ XZe \end{bmatrix} = 0. \tag{2.3}$$

A solução do sistema será a solução dos problemas primal e dual de programação linear. Este sistema pode ser resolvido pelo método de Newton (Ruggiero & Lopes, 1996).

Para isto, considere o vetor de resíduos definido pela Equação (2.4).

$$-F(x^0, y^0, z^0) = \begin{bmatrix} -Ax^0 + b \\ -A^t y^0 - z^0 + c \\ -X^0 Z^0 e \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_a \end{bmatrix} = r^k \tag{2.4}$$

A partir do ponto inicial (x^0, y^0, z^0), considerando um passo $\alpha = 1$, o próximo ponto obtido pelo método de Newton será:

$$(x, y, z) = (x^0, y^0, z^0) - J^{-1}(x^0, y^0, z^0)F(x^0, y^0, z^0)$$

$$(x, y, z) = (x^0, y^0, z^0) + \alpha(\Delta x, \Delta y, \Delta z)$$

$$(\Delta x, \Delta y, \Delta z) = -J^{-1}(x^0, y^0, z^0)F(x^0, y^0, z^0)$$

em que $J(x^0, y^0, z^0)$ é o jacobiano do sistema (2.3), definido como segue:

$$J(x^0, y^0, z^0) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^0 & 0 & X^0 \end{bmatrix}.$$

O novo ponto $(x^{k+1}, y^{k+1}, z^{k+1})$ é obtido como:

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha(\Delta x^k, \Delta y^k, \Delta z^k).$$

A direção de Newton $(\Delta x^k,\ \Delta y^k,\ \Delta z^k)$ é calculada por meio da solução do sistema linear definido pela Equação (2.5).

$$J(x^k, y^k, z^k)(\Delta x^k, \Delta y^k, \Delta z^k) = -F(x^k, y^k, z^k)$$

$$J(x^k, y^k, z^k)(\Delta x^k, \Delta y^k, \Delta z^k) = r^k$$

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_a^k \end{bmatrix}. \tag{2.5}$$

Nos métodos de pontos interiores, a interioridade do ponto ($x > 0$ e $z > 0$) é garantida por meio do passo α. Com isto o tamanho do passo α^k será calculado como estabelecido pela Equação (2.6).

$$\begin{aligned} \alpha_p^k &= min\left[1, \tau \cdot min\left(\frac{-x_i^k}{\Delta x_i^k}\right)\right] \text{ tal que } \Delta x_i^k < 0 \\ \alpha_d^k &= min\left[1, \tau \cdot min\left(\frac{-z_i^k}{\Delta z_i^k}\right)\right] \text{ tal que } \Delta z_i^k < 0 \end{aligned} \tag{2.6}$$

em que $\tau \in (0,1)$.

A aplicação do método de Newton às condições de otimalidade descreve o método primal-dual afim, resumido no Procedimento 2.1.

Procedimento 2.1: Método Primal-Dual Afim

Entrada: $x^0 \in \mathbb{R}^n, z^0 \in \mathbb{R}^n, (x^0, z^0) > 0, y^0 \in \mathbb{R}^m,$

Para $k = 0, 1, 2 \ldots$ **faça**

1. Calcular r_p^k, r_d^k, r_a^k (Equação (2.4))
2. Calcular $(\Delta x^k, \Delta y^k, \Delta z^k)$ (Equação (2.5))
3. Calcular o tamanho do passo α^k (Equação (2.6))
4. Calcular o novo ponto:

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta z^k)$$

Fim.

2.3.2. Método primal-dual seguidor de caminho

Nos métodos primais duais afim escala, os pontos (x, z) se aproximam muito rápido dos seus limites e os pontos obtidos nas iterações seguintes tendem a tangenciar a fronteira, provocando problemas de convergência do método. Conforme o método progride, os pontos precisam manter uma trajetória central na região de factibilidade. Para isto, os termos $x_i z_i$ devem convergir para zero, na mesma velocidade.

Se uma perturbação μ é introduzida na condição de complementariedade como mostra a Equação 2.7, os pontos x e z irão manter a trajetória central.

$$x_i z_i = \mu^k. \tag{2.7}$$

A perturbação μ^k é inicializada com um valor alto que vai diminuindo à medida que o método se aproxima de uma solução do problema; $\mu^k = 0$ quando $k \to \infty$. Seguindo esta heurística, o valor de μ^k será calculado como:

$$\mu^k = \frac{(x^k)^t z^k}{n}. \tag{2.8}$$

O novo sistema é descrito pela Equação (2.9)

$$-F(x^k, y^k, z^k) = \begin{bmatrix} -Ax^k + b \\ -A^t y^k - z^k + c \\ \mu^k e - X^k Z^k e \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_{ce}^k \end{bmatrix}. \quad (2.9)$$

O sistema para calcular as direções é equacionado com a Equação (2.10).

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_{ce}^k \end{bmatrix}. \quad (2.10)$$

O método primal-dual seguidor de caminho pode ser resumido como descreve o Procedimento 2.2:

Procedimento 2.2: Método Primal-Dual Seguidor de Caminho

Entrada: $x^0 \in \mathbb{R}^n, z^0 \in \mathbb{R}^n, (x^0, z^0) > 0, y^0 \in \mathbb{R}^m$

Para $k = 0, 1, 2 \ldots$ **faça**

1. Calcular μ^k (Equação (2.8))
2. Calcular r_p^k, r_d^k, r_c^k (Equação (2.4))
3. Calcular $(\Delta x^k, \Delta y^k, \Delta z^k)$ (Equação (2.5))
4. Calcular o tamanho do passo α^k (Equação (2.6))
5. Calcular o novo ponto:

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha^k (\Delta x^k, \Delta y^k, \Delta z^k)$$

Fim.

2.3.3. Método preditor-corretor

O método preditor-corretor (Mehrotra, 1992) é uma das variações mais implementadas do método primal-dual e com melhores resultados na solução de problemas com estruturas genéricas. Neste método, a direção de busca é formada por dois componentes:

1. Direção preditora ou afim $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{z})$ sem perturbação, definida pela Equação (2.11).

$$J(x^k)\begin{bmatrix}\Delta\tilde{x}^k\\ \Delta\tilde{y}^k\\ \Delta\tilde{z}^k\end{bmatrix} = \begin{bmatrix} b - Ax^k \\ -A^t y^k - z^k + c \\ -X^k Z^k e\end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_a^k\end{bmatrix}. \quad (2.11)$$

O tamanho do passo nesta direção será o máximo que permita manter o ponto interior:

$$\tilde{\alpha}_p^k = argmax\{\alpha \in [0,1] | x^k + \alpha\Delta\tilde{x}^k > 0\},$$

$$\tilde{\alpha}_d^k = argmax\{\alpha \in [0,1] | z^k + \alpha\Delta\tilde{z}^k > 0\}.$$

Na Equação (2.12), é definido o gap de dualidade afim $\tilde{\mu}^k$, em que é medida a eficiência da direção preditora:

$$\tilde{\mu}^k = \frac{(x^k + \tilde{\alpha}_p^k\Delta\tilde{x}^k)(z^k + \tilde{\alpha}_d^k\Delta\tilde{z}^k)}{n}. \quad (2.12)$$

A partir de $\tilde{\mu}^k$ (Equação (2.12)) é calculado o parâmetro de centragem σ. Mehrotra (Mehrotra, 1992) sugere a heurística formulada pela Equação (2.13):

$$\sigma^k = \left(\frac{\tilde{\mu}^k}{\mu^k}\right)^3. \quad (2.13)$$

em que μ^k é definida na Equação (2.8).

2. Direção corretora $(\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{z})$. Esta direção usa o parâmetro de centragem σ (Equação (2.13)). É calculada a partir do sistema (2.14) que possui a mesma matriz dos coeficientes do sistema (2.11).

$$J(x^k)\begin{bmatrix}\Delta\hat{x}^k\\ \Delta\hat{y}^k\\ \Delta\hat{z}^k\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma^k\mu^k e - \Delta\tilde{X}^k\Delta\tilde{Z}^k e\end{bmatrix}. \quad (2.14)$$

Finalmente, a direção de busca é calculada como a soma da direção afim com a direção corretora:

$$(\Delta x^k, \Delta y^k, \Delta z^k) = (\Delta\tilde{x}^k, \Delta\tilde{y}^k, \Delta\tilde{z}^k) + (\Delta\hat{x}^k, \Delta\hat{y}^k, \Delta\hat{z}^k).$$

No método preditor-corretor, a cada passo, é necessário resolver dois sistemas de equações lineares com a mesma matriz de coeficientes. Embora a resolução desses sistemas seja a fase de maior custo computacional na resolução do problema, o método ainda se torna viável por gerar uma redução significativa na quantidade de iterações necessárias para a convergência (Vanderbei, 2014).

2.3.4. Cálculo das direções

O passo de maior custo computacional em métodos de pontos interiores é o cálculo das direções $(\Delta x^k, \Delta y^k, \Delta z^k)$. No método primal dual seguidor de caminho (Seção 2.3.2), a direção é calculada a partir de um único sistema (2.10). O método preditor corretor calcula as direções solucionando dois sistemas lineares (2.11) e (2.14) com a mesma matriz dos coeficientes. De forma geral o sistema linear a ser resolvido é equacionado como Equação (2.15):

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_c^k \end{bmatrix}. \tag{2.15}$$

A matriz do sistema em (2.15) é muito esparsa; exceto os blocos A e A^t, todos os blocos são diagonais ou zeros. A matriz do sistema (2.15) pode ser reduzida ao sistema (2.16) chamado de sistema aumentado, através da eliminação de variáveis:

$$\begin{bmatrix} -\theta^{-1} & A^t \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \end{bmatrix} = \begin{bmatrix} r_d^k - X^{-1} r_c^k \\ r_p^k \end{bmatrix}, \tag{2.16}$$

em que $\theta = Z^{-1}X$. A direção Δz^k é calculada como: $\Delta z^k = X^{-1}(r_c^k - Z\Delta x^k)$. Este sistema pode ser mais compactado se a variável Δx^k for eliminada. O novo sistema obtido é apresentado na Equação (2.17):

$$(A\theta A^t)\Delta y^k = r_p^k + A\theta\left(r_d^k - X^{-1} r_c^k\right). \tag{2.17}$$

O valor de Δx^k será calculado como $\Delta x^k = \theta A^t \Delta y^k - \theta\left(r_d^k - X^{-1} r_c^k\right)$.

O sistema representado na Equação (2.17) é um sistema de equações normais (Golub & Loan, 2013). Neste tipo de sistema, a matriz dos coeficientes possui

características especiais, o que permite a aplicação de técnicas específicas para a sua resolução.

Especificamente, a matriz $A\theta A^t$ é de dimensão m (número de restrições). A matriz diagonal θ é de dimensão n e os elementos da diagonal são $\theta_i = z_i^k \left(x_i^k\right)^{-1} > 0$, pois $z_i^k > 0$ e $x_i^k > 0$. Considerando que matriz A do problema de programação linear é de posto completo e os elementos de θ são positivos, então a matriz $A\theta A^t$ será simétrica e definida positiva.

Adicionalmente, o padrão de esparsidade desta matriz não varia durante todo o processo iterativo. Isto é, a quantidade e a posição dos elementos não nulos serão as mesmas em todas as iterações, pois embora a matriz θ varie a cada iteração, todos seus elementos são positivos. Esta vantagem é muito significativa no uso de método de reordenação. Somente será necessária uma única reordenação da matriz antes do processo iterativo.

No capítulo que segue serão discutidos os métodos de solução dos sistemas lineares (2.17).

3. SISTEMAS LINEARES

Os métodos de solução para sistemas lineares são divididos em dois grupos, os métodos diretos e os métodos iterativos (Ruggiero & Lopes, 1996). Os métodos diretos fornecem soluções exatas, salvo erros de arredondamento numéricos, com um número limitado e conhecido de passos. Os métodos iterativos são projetados para obter valores próximos à solução exata do sistema linear atendendo os critérios de parada, não tendo um valor previamente definido do número de iterações que serão necessárias para chegar à solução.

No método preditor-corretor de pontos interiores, as direções de busca são obtidas a partir da solução de dois sistemas lineares que compartilham a mesma matriz dos coeficientes. Estes sistemas, de forma geral, podem ser expressos como mostra a Equação (3.1).

$$(A\theta A^t)\Delta y = r \tag{3.1}$$

A matriz dos coeficientes $A\theta A^t$ é quase sempre muito esparsa e o padrão da esparsidade é conservado durante todo o processo iterativo.

Já o condicionamento de tal matriz varia muito; ela se torna cada vez mais mal condicionada à medida que o método de pontos interiores se aproxima de uma solução ótima. O deterioramento do condicionamento é provocado pela matriz θ. Os elementos da diagonal desta matriz são calculados como: $\theta_i = x_i^k \left(z_i^k\right)^{-1}$ e se a condição de complementariedade for considerada (condição 3 de otimalidade), tem-se:

$$\text{Se } x_j^* > 0 \ \rightarrow \ z_j^* = 0,$$

$$\text{Se } z_j^* > 0 \ \rightarrow \ x_j^* = 0.$$

Consequentemente, os valores θ_i serão muito grandes ou muito pequenos, o que provoca o mal condicionamento da matriz.

Para a solução destes sistemas podem ser usados métodos diretos ou métodos iterativos. A matriz dos coeficientes é simétrica e definida positiva (Seção 2.3.4); o método direto apropriado é a fatoração de Cholesky (Golub & Loan, 2013) e o método

iterativo é o método dos gradientes conjugados (Saad, 2003). A seguir, são descritos estes métodos.

3.1. Fatoração de Cholesky

Considere o sistema $Gw = d$, em que $G \in \mathbb{R}^{n \times n}$ é uma matriz simétrica definida positiva e sejam $w, d \in \mathbb{R}^n$. A matriz G pode ser decomposta como: $G = LL^t$ onde L é uma matriz triangular inferior. Esta decomposição é conhecida como decomposição de Cholesky (Golub & Loan, 2013) e o teorema a seguir garante a sua existência.

Teorema 1. Se $G \in \mathbb{R}^{n \times n}$ é simétrica e definida positiva, então existe uma única matriz triangular inferior $L \in \mathbb{R}^{n \times n}$, em que os elementos da diagonal principal são todos positivos, tal que $G = LL^t$ (Golub & Loan, 2013).

Para a solução de $(A\theta A^t)\Delta y = r$ utiliza-se a decomposição como segue:

$$(A\theta A^t)\Delta y = r \rightarrow (LL^t)\Delta y = r$$

Com isso, a solução Δy é calculada através da solução dos sistemas triangulares que seguem:

$$Lv = r$$
$$L^t \Delta y = v$$

O vetor v é facilmente calculado a partir de substituições sucessivas no sistema triangular inferior ($Lv = r$). Após a obtenção do valor de v, o valor de Δy é calculado por substituições retroativas no sistema triangular superior ($L^t \Delta y = v$).

A matriz $A\theta A^t$ dos sistemas de métodos de pontos interiores é geralmente muito esparsa. A fatoração de Cholesky, sem estratégias adequadas, produz fatores com preenchimento maior que aqueles da matriz original. Em problemas esparsos de grande porte, o uso deste método de solução é proibitivo.

3.2. Método dos gradientes conjugados

A limitação discutida anteriormente na solução dos sistemas pela fatoração de Cholesky pode ser contornada quando são utilizados métodos iterativos.

O método iterativo dos gradientes conjugados é utilizado para a solução de sistemas lineares com matrizes simétricas e definidas positivas (Saad, 2003).

Suponha o sistema $Gw = d$; seja $G \in \mathbb{R}^{n \times n}$ uma matriz simétrica definida positiva e sejam $w, d \in \mathbb{R}^n$. Seja f uma função quadrática $f: \mathbb{R}^{n \times n} \to \mathbb{R}$, definida como na Equação (3.2).

$$f(w) = \frac{1}{2} w^t G w - d^t w + q, \tag{3.2}$$

em que q é um valor constante.

A partir de um ponto w arbitrário, o vetor de gradiente aponta na direção de maior crescimento de $f(w)$. Resolvendo $\nabla f(w) = Gw - d = 0$, encontra-se um ponto crítico de $f(w)$. A matriz G é simétrica definida positiva consequentemente a inversa existe e, com isso, o sistema terá solução única. Então, resolver o sistema $Gw = d$ é equivalente a encontrar w que minimiza $f(w)$.

O método de otimização não linear dos gradientes conjugados é utilizado para solucionar o problema de minimização de $f(w)$ e, com isto, o sistema $Gw = d$ será solucionado. Neste método, as direções de busca são construídas por conjugação de resíduos. A descrição do método pode ser encontrada em (Saad, 2003).

A convergência do método dos gradientes conjugados depende diretamente dos autovalores da matriz do sistema. Se os autovalores tiverem valores aproximados, o método irá convergir mais rapidamente. Quanto maior a diferença entre os autovalores, maior será o número de iterações necessário para a convergência. Um sistema resolvido através do método dos gradientes conjugados será resolvido com, no máximo, um número de iterações igual ao número de autovalores diferentes que a matriz possui (Saad, 2003).

O mal condicionamento da matriz pode ser contornado com o uso de precondicionadores. Bocanegra (Bocanegra, 2005) propôs um precondicionador híbrido para o condicionamento da matriz $A\theta A^t$ dos métodos de pontos interiores.

3.3. Precondicionador híbrido

O termo precondicionamento se refere à transformação de um sistema mal condicionado em outro sistema com condições mais favoráveis à solução, por meio do uso de métodos iterativos. Um precondicionador é uma matriz com esse efeito de transformação. Em problemas que envolvem matriz simétrica definida positiva, a convergência do método dos gradientes conjugados depende da distribuição dos autovalores de G (Benzi, 2002). O precondicionamento busca uma nova matriz mais próxima da matriz identidade ou, então, com autovalores mais próximos da unidade.

Suponha o sistema de equações lineares $Gw = d$ e seja M o precondicionador. Uma técnica de precondicionamento de um sistema é a multiplicação pela esquerda e pela direita da matriz do sistema pela matriz M^{-1} e M^{-t} respectivamente:

$$M^{-1}GM^{-t}z = M^{-1}d; w = M^{-t}z \tag{3.3}$$

Um precondicionador deve ser de fácil construção, baixo custo, e com esparsidade próxima à da matriz dos coeficientes G.

Em (Bocanegra, 2005) foi proposto um precondicionador híbrido, para matrizes $A\Theta A^t$ oriundas de métodos de pontos interiores. O precondicionador híbrido utiliza dois precondicionadores diferentes em fases diferentes do processo de otimização. Nas iterações iniciais ou primeira fase, é utilizado um precondicionador baseado na fatoração incompleta de Cholesky; a fatoração controlada de Cholesky (Campos & Birkett, 1998). Nas iterações finais ou segunda fase utiliza-se um precondicionador separador (Oliveira & Sorensen, 2005) específico para matrizes mal condicionadas. A mudança de precondicionador é determinada por uma heurística de mudança de fase proposta em (Velazco et al., 2010).

Nas seções seguintes, serão discutidos os precondicionadores que formam o precondicionador híbrido, assim como a heurística de mudança de fase.

3.3.1. Fatoração Controlada de Cholesky

A fatoração controlada de Cholesky (FCC) (Campos & Birkett, 1998) é uma fatoração incompleta de Cholesky em que o preenchimento do fator é controlado por um parâmetro (η) que depende do espaço de memória disponível.

Suponha a matriz dos coeficientes $A\Theta A^t$ fatorada como mostra a Equação (3.4):

$$A\Theta A^t = LL^t = \tilde{L}\tilde{L}^t + R \tag{3.4}$$

sendo:

$L =$ Fator obtido pela fatoração de Cholesky;

$\tilde{L} =$ Fator obtido pela fatoração incompleta de Cholesky;

$R =$ Matriz de resíduo.

Seja $\tilde{L}$ a matriz precondicionadora. A matriz $A\Theta A^t$ precondicionada será:

$$T = \tilde{L}^{-1}\mathrm{A\Theta A^t}\tilde{L}^{-t} = (\tilde{L}^{-1}L)(\tilde{L}^{-1}L)^t\,. \tag{3.5}$$

Define-se $E = L - \tilde{L}$. Substituindo $L = E + \tilde{L}$ na Equação (3.5), tem-se:

$$T = \tilde{L}^{-1}\mathrm{A\Theta A^t}\tilde{L}^t = (I + \tilde{L}^{-1}E)(I + \tilde{L}^{-1}E)^t\,.$$

Com isso pode-se notar que, quando $\tilde{L} \to L$ (o fator incompleto $\tilde{L}$ mais próximo do fator completo L) tem-se $E \to 0$ (a diferença entre eles tende a zero) e consequentemente, $\tilde{L}^{-1}\mathrm{A\Theta A^t}\tilde{L}^t \to I$ (a matriz precondicionada T será mais próxima da identidade).

Quanto mais próxima a matriz precondicionada T for da matriz identidade, melhor será a convergência do GCP. Portanto, o número de iterações do GCP está diretamente relacionado com a norma da matriz de resíduo R. A matriz R pode ser expressa como na Equação (3.6):

$$R = LE^t + E\tilde{L}^t. \tag{3.6}$$

A FCC é baseada na minimização da norma de Frobenius de E, pois quando $\|E\| \to 0$, implicará $\|R\| \to 0$. Considere o problema equacionado na Equação (3.7):

$$minimizar\ ||E||_F^2 = \sum_{j=1}^{m} c_j, \tag{3.7}$$

em que $c_j = \sum_{i=1}^{m} | l_{ij} - \tilde{l}_{ij}|^2$. Separa-se c_j em dois somatórios e obtém-se:

$$c_j = \sum_{k=1}^{m_j+\eta} |l_{i_k j} - \tilde{l}_{i_k j}|^2 + \sum_{k=m_j+\eta+1}^{m} | l_{i_k j}|^2,$$

sendo que m_j representa a quantidade de elementos não nulos abaixo da diagonal da coluna j da matriz original ($A\Theta A^t$) e η representa o número de elementos não nulos que podem ser adicionados a cada coluna durante a FCC.

No primeiro somatório estão os $m_j + \eta$ elementos não nulos da j-ésima coluna de $\tilde{L}$. No segundo somatório, estão apenas os elementos de L que não estão em $\tilde{L}$.

Dessa forma, o problema de minimização de E pode ser resolvido usando a heurística:

- Aumentar o parâmetro η, permitindo maior preenchimento por coluna. Com isso c_j diminui, pois o primeiro somatório contém mais elementos.
- Escolher os $m_j + \eta$ maiores elementos de $\tilde{L}$ em valor absoluto para um η fixo. Assim, os maiores elementos ficam no primeiro somatório e os menores no segundo, produzindo um fator $\tilde{L}$ mais próximo ao original para a quantidade determinada de armazenamento.

O preenchimento inicial η depende da média da quantidade de elementos não nulos da matriz $A\Theta A^t$.

O preenchimento η do fator $\tilde{L}$ varia até atingir um valor máximo. Este aumento está relacionado à convergência do GCP: se o GCP convergir em um número de iterações superior a $\frac{m}{6}$, sendo m a ordem de $A\Theta A^t$, então $\eta = \eta + 10$. O valor máximo é definido a partir do problema.

3.3.2. Troca de precondicionadores

O preenchimento do fator $\tilde{L}$ pode atingir seu valor máximo quando o GCP não converge à solução do sistema. Neste caso, o FCC não está efetuando um bom

precondicionamento do sistema e é realizada a troca de precondicionadores ou troca de fases.

A troca de precondicionadores é realizada quando o valor de η atinge o máximo e as iterações do GCP são superiores à $\frac{m}{6}$ (Velazco et al., 2010). O valor máximo para o parâmetro η depende do problema; no Capítulo 5 serão apresentados os valores utilizados nos experimentos conduzidos.

Em alguns problemas a troca de fases não acontece e o FCC é utilizado durante todo o processo iterativo.

3.3.3. Precondicionador separador

O precondicionador separador foi definido por (Oliveira & Sorensen, 2005) para os sistemas aumentados (Capítulo 2, Equação (2.16)). Neste trabalho, uma versão deste precondicionador para a matriz $A\Theta A^t$ foi utilizada.

Considere a matriz $A = [B\ N]P$ ordenada por colunas a partir de uma matriz de permutação $P \in \mathbb{R}^{n \times n}$ tal que, $B \in \mathbb{R}^{m \times m}$ é uma matriz não singular e $N \in \mathbb{R}^{n \times (n-m)}$ são as colunas restantes. Dessa forma, $A\Theta A^t$ pode ser representada como na Equação (3.8):

$$A\Theta A^t = [B\ N]P^t\Theta P\begin{bmatrix}B^t\\N^t\end{bmatrix} = [B\ N]\begin{bmatrix}\Theta_B & 0\\0 & \Theta_N\end{bmatrix}\begin{bmatrix}B^t\\N^t\end{bmatrix} = B\Theta_B B^t + N\Theta_N N^t. \tag{3.8}$$

A Equação (3.8) pode ser multiplicada por $\Theta_B^{-1/2}B^{-1}$ e pós multiplicada por $\left(\Theta_B^{-1/2}B^{-1}\right)^t$ resultando na matriz T, como mostra a Equação (3.9):

$$T = \Theta_B^{-1/2}B^{-1}(A\Theta A^t)B^{-t}\Theta_B^{-1/2} = I_m + WW^t, \tag{3.9}$$

sendo $W = \Theta_B^{-1/2}B^{-1}N\Theta_N^{-1/2} \in \mathbb{R}^{m\ X\ (n-m)}$.

Próximo a uma solução ótima do problema de programação linear, ao menos $n - m$ elementos de Θ^{-1} serão grandes. Uma escolha adequada das colunas de B resultará

elementos da diagonal de $\theta_B^{-1/2}$ e θ_N muito pequenos. Consequentemente, a matriz W se aproximará à identidade e a matriz precondicionada T terá autovalores perto da unidade o que acelerará a convergência do gradiente conjugado.

O passo computacionalmente mais caro na construção do precondicionador separador é o de encontrar a matriz não singular B. Em (Velazco et al., 2010), são selecionadas as primeiras m colunas linearmente independentes de $A\theta$ com maior norma *2*. Para isto, primeiramente são ordenadas as colunas de $A\theta$; na sequência é feita a fatoração LU para encontrar as m colunas linearmente independentes. As colunas selecionadas podem ser usadas por várias iterações para a construção do precondicionador; o reuso das mesmas dependerá do desempenho do GCP. Quando as colunas selecionadas são reusadas, a construção do precondicionador separador será muito barata, pois somente a matriz diagonal θ muda a cada iteração.

4. MÉTODOS DE REORDENAÇÃO

A fatoração de Cholesky aplicada em matrizes esparsas aumenta a quantidade de elementos não nulos (George, et al., 1994), gerando um maior custo computacional para armazenamento e processamento da matriz.

Suponha uma matriz G simétrica de dimensão $m \times m$ e suponha um vetor de dimensão m com os índices das linhas/colunas da matriz ordenados de 1 até m. A reordenação de uma matriz resultará em um novo vetor sendo que, os índices das linhas/colunas terão uma ordem diferente ao original seguindo os critérios da estratégia de reordenação escolhida.

Algebricamente, a reordenação de uma matriz é gerada a partir da multiplicação da matriz original G por uma matriz de permutação P, em que P é obtida por meio da construção da matriz identidade com as linhas alteradas conforme o método de reordenação escolhido. Ao aplicar PGP^t (multiplicar a matriz G pela esquerda e pela direita pela matriz de permutação P e P^t respectivamente) obtém-se uma matriz reordenada com as mesmas linhas/colunas de G, porém com as posições de suas linhas/colunas alteradas.

O objetivo de reordenar uma matriz esparsa é reduzir o efeito de preenchimento excessivo que ocorre durante o processo de fatoração (George, et al., 1994). Obter a reordenação de uma matriz esparsa que provoque menor preenchimento é um problema NP-Completo (Yannakakis, 1981).

Em matrizes não esparsas não há necessidade de aplicar métodos de reordenação, pois sua estrutura já é densa e, portanto, não sofrerá um grande aumento de valores não nulos em relação à sua estrutura original, durante o processo de fatoração de Cholesky.

Carmo (Carmo, 2005) avaliou a influência de três métodos de reordenação na resolução de sistemas lineares com o GCP com o precondicionador FCC. A pesquisa realizada concluiu que a reordenação da matriz dos coeficientes pelo MG melhora a qualidade do FCC. Quando a matriz dos coeficientes é reordenada, o precondicionador FCC tende a ser mais próximo da fatoração completa o que melhora o precondicionamento da matriz.

Em métodos de pontos interiores de tipo primal-dual a cada iteração são resolvidos um ou dois sistemas lineares com a mesma matriz dos coeficientes; $A\Theta A^t$. Esta matriz, como foi discutido no Capítulo 3 é simétrica e o mesmo padrão de esparsidade é mantido durante todo o processo de otimização (Θ muda a cada iteração, mas seus elementos são sempre maiores que zero). Isto facilita o uso de métodos de reordenação; são usados somente uma única vez antes do processo de otimização. Adicionalmente, reordenar $A\Theta A^t$ é equivalente a reordenar a matriz A por linhas.

Existem diversos métodos de reordenação para matrizes (Sloan, 1986) (Cuthill & McKee, 1969) (Amestoy, et al., 1996) (George, 1973). Os métodos de reordenação MG e ND são estratégias que representam a matriz através grafos. A próxima seção introduz alguns conceitos de Teoria dos Grafos relevantes para um melhor entendimento dos métodos de reordenação em matrizes esparsas.

4.1. Conceitos básicos de Teoria dos Grafos

Um grafo $G = (V, E)$ finito é composto por dois conjuntos finitos V e E:

- $V = \{v_1, v_2, \dots, v_n\}$ é o conjunto não vazio de vértices do grafo em que n representa o número de vértices;
- $E = \{e_1, e_2, \dots, e_z\}$ é o conjunto de arestas, com seus elementos denotados por e_i em que z representa o número de arestas. Cada aresta e_i é um par não ordenado de vértices de V.

Grafos podem ser representados por diagramas, onde os vértices são representados por pontos e as arestas são representadas por linhas.

A seguir são definidos os conceitos importantes para o trabalho (Nicoletti & Hruschka, 2006):

- Vértices adjacentes: Vértices que estão unidos pela mesma aresta.
- Arestas adjacentes: Arestas que tem um vértice em comum.
- Grau de um vértice v: É o número de arestas de G que são incidentes em v.

- Sejam dois grafos $G1 = (V1, E1)$ e $G2 = (V2, E2)$. Diz-se que $G2$ é subgrafo de $G1$ se $V2 \subseteq V1$ e $E2 \subseteq E1$ e, para toda aresta $e \in E2$, se e for incidente a $v1$ e $v2$, então $v1, v2 \in V2$.
- Um grafo $G = (V, E)$ é chamado bipartido quando V puder ser particionado em dois conjuntos não vazios, X e Y, em que $X \cup Y = V$ e $X \cap Y = \emptyset$ e, além disso, se $e = (x, y) \in E$ necessariamente $x \in X$ e $y \in Y$. A partição $V = X \cup Y$ é chamada bipartição de G.

Uma matriz simétrica A de dimensão n pode ser representada por um grafo $G = (V, E)$, em que $|V| = n$ e o conjunto E terá um elemento conectando v_i e v_j para cada $a_{ij} \neq 0$ com $i \neq j$. Cada vértice do grafo representa uma linha/coluna da matriz. As arestas do grafo indicam a relação existente entre as linhas e colunas da matriz, sendo que se tivermos um elemento não nulo na posição a_{ij} da matriz, teremos uma aresta ligando o vértice v_i ao vértice v_j no grafo. Os elementos da diagonal da matriz, a_{ii}, não serão considerados na representação, o grafo não terá laços.

A Tabela 4.1 mostra uma matriz simétrica de dimensão $n = 4$ e a Figura 4.1, o grafo que representa esta matriz. Na Tabela 4.1, as linhas e colunas são indicadas através de números e cada elemento não nulo é representado por X. O valor de a_{ij} é irrelevante no processo de reordenação.

Tabela 4.1 - Matriz simétrica de dimensão 4

$\boldsymbol{A}$ =

	1	**2**	**3**	**4**
1	X	X		X
2	X	X	X	
3		X	X	X
4	X		X	X

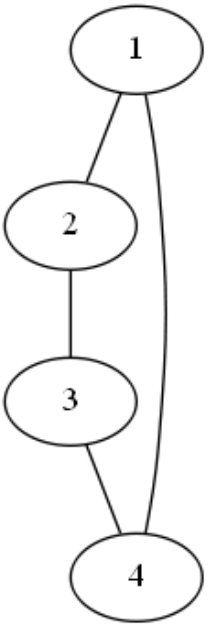

Figura 4.1 –Grafo da matriz *A*

Observe que na matriz da Tabela 4.1, os elementos a_{12} e a_{14} são não nulos, consequentemente, o vértice 1 será conectado com os vértices 2 e 4 através de arestas. Seguindo este raciocínio podem ser identificadas todas as arestas do grafo sendo construído.

O processo de reordenação de uma matriz pode ser visto como a reordenação dos vértices de um grafo. No exemplo anterior, seja uma ordenação inicial $O_0 = [1, 2, 3, 4]$, em que cada vértice representa uma linha da matriz A. Suponha uma reordenação dos vértices $O_1 = [4, 2, 3, 1]$; o grafo obtido com a reordenação é representado na Figura 4.2 e a matriz reordenada, na Tabela 4.2.

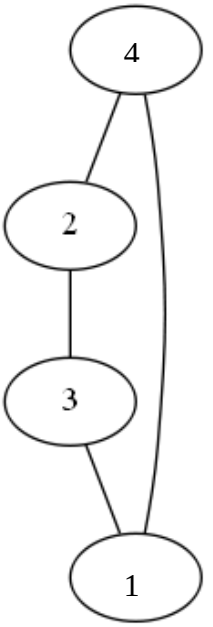

Figura 4.2 - Grafo da matriz A reordenada

Tabela 4.2 - Matriz simétrica de dimensão 4 reordenada

A **=**

	1	**2**	**3**	**4**
1	X		X	X
2		X	X	X
3	X	X	X	
4	X	X		X

4.2. Heurística de Mínimo Grau

A heurística MG (George, et al., 1994) é a mais utilizada para a reordenação de matrizes esparsas. Nesta heurística de reordenação dos vértices, o vértice inicial será o vértice de menor grau.

O vértice de menor grau no grafo será a linha/coluna que tem a maior quantidade de elementos nulos. A reordenação acontece permutando a linha/coluna menos densa com a primeira posição que ainda não tenha sido alterada da matriz. Uma linha/coluna já reordenada não será mais considerada no processo de reordenação. Logo, na representação da matriz como um grafo, o vértice permutado é eliminado do grafo. A eliminação do vértice implicará que seus vértices adjacentes se tornem adjacentes entre

eles (se eles não forem antes da eliminação) e com isto, novas arestas serão criadas. A criação de novas arestas provocará novos preenchimentos na matriz após a fatoração.

O processo de permutação irá acontecer de maneira sequencial, a partir do vértice de menor grau para o de maior grau, não levando em consideração o preenchimento provocado depois da adição de novas arestas. Com isso, o método de reordenação MG pode ser categorizado como um algoritmo guloso.

Depois de n passos, o resultado da reordenação será uma matriz em que nas primeiras posições estarão as linhas/colunas com maior esparsidade e as mais densas no final.

Procedimento 4.1: Mínimo Grau

Entrada: Grafo G de $\boldsymbol{A}$

Saída: Vetor de vértices reordenados de G, $z = [z_1, z_2, \cdots, z_n]$

Criar $\boldsymbol{z}$ vazio

$k = 1$

Enquanto $k \leq n$ **faça**

1. Escolher o vértice de menor grau v_k do grafo G
2. Se os vértices adjacentes a v_k não forem adjacentes então criar novas arestas entre eles.
3. Eliminar vértice de v_k de G
4. Fazer $z_k = v_k$
5. $k = k + 1$

Fim Enquanto

Retorna z

Fim.

O funcionamento do MG pode ser verificado no exemplo a seguir.

Suponha a matriz A simétrica de dimensão 10 representada na Tabela 4.3. Da diagonal para cima da matriz há 31 elementos nulos. A melhor reordenação possível para esta matriz seria uma nova matriz que quando fatorada tenha a mesma quantidade de elementos não nulos que a matriz original.

Tabela 4.3 - Matriz *A* de dimensão 10

	1	2	3	4	5	6	7	8	9	10
1	1000	0	0	14	0	0	0	0	0	0
2	0	2000	23	24	25	0	0	28	29	0
3	0	23	3000	0	35	36	37	38	39	0
4	14	24	0	4000	0	0	0	48	0	0
5	0	25	35	0	5000	0	0	0	0	0
6	0	0	36	0	0	6000	0	0	0	0
7	0	0	37	0	0	0	7000	78	79	0
8	0	28	38	48	0	0	78	8000	0	0
9	0	29	39	0	0	0	79	0	9000	0
10	0	0	0	0	0	0	0	0	0	10000

Suponha que a matriz A é fatorada pela fatoração de Cholesky obtendo $A = LL^t$. Na Tabela 4.4, é apresentado o fator L^t obtido na fatoração, note que possui 18 elementos nulos. O fator tem mais elementos não nulos que a matriz original no qual será necessária mais memória para seu armazenamento.

Tabela 4.4 – Fator L^t obtido após a fatoração por Cholesky da matriz *A*

316.228	0	0	0,4427	0	0	0	0	0	0
0	447.214	0,5143	0,5367	0,5590	0	0	0,6261	0,6485	0
0	0	547.698	-0,0050	0,6338	0,6573	0,6756	0,6879	0,7060	0
0	0	0	632.417	-0,0047	0,0001	0,0001	0,7537	-0,0054	0
0	0	0	0	707.056	-0,0059	-0,0061	-0,0111	-0,0115	0
0	0	0	0	0	774.569	-0,0057	-0,0058	-0,0060	0
0	0	0	0	0	0	836.633	0,9268	0,9386	0
0	0	0	0	0	0	0	894.299	-0,0197	0
0	0	0	0	0	0	0	0	948.588	0
0	0	0	0	0	0	0	0	0	1.000.000

A reordenação da matriz A pelo MG provocará menor preenchimento após a fatoração.

Para a reordenação da matriz A pelo MG é necessária sua representação em um grafo como apresentado na Figura 4.3.

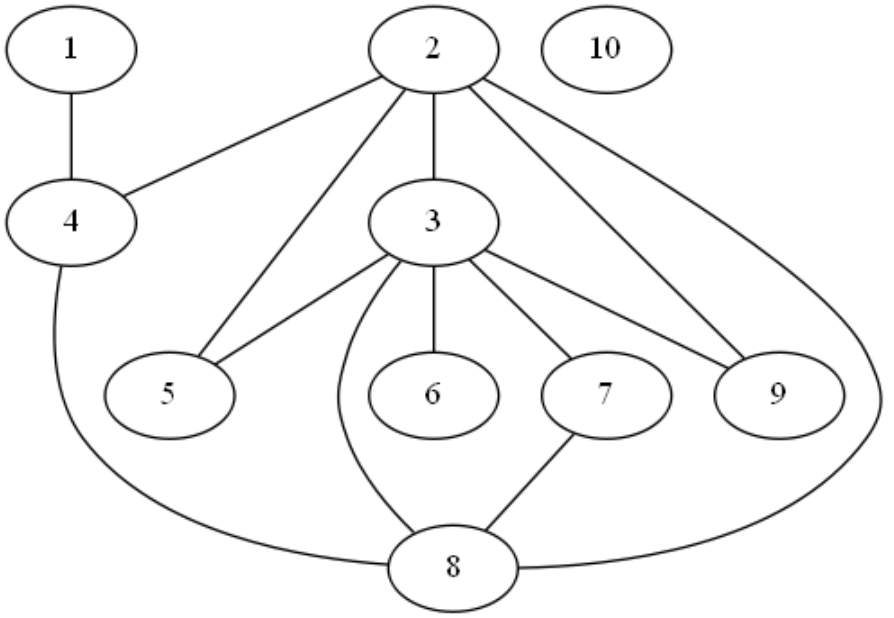

Figura 4.3 – Grafo G criado a partir da matriz A

A Figura 4.4 mostra o processo de eliminação dos vértices de menor grau do grafo *G*.

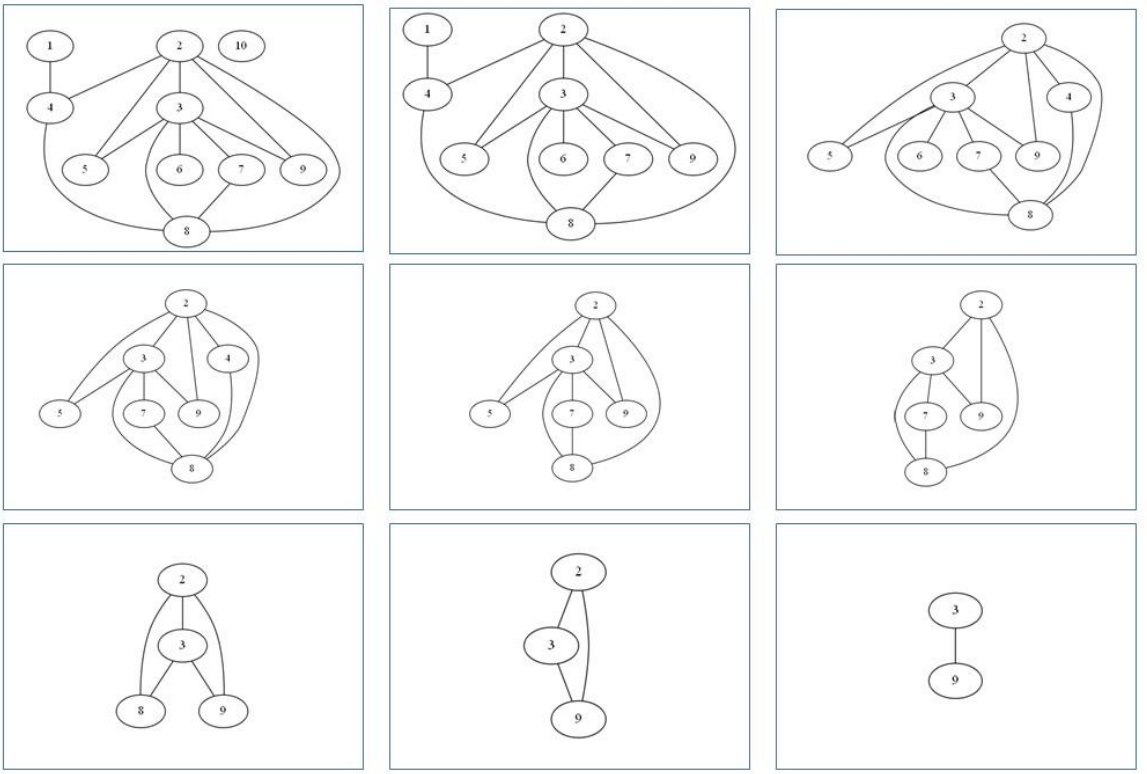

Figura 4.4 - Eliminação dos vértices de menor grau do grafo G pelo MG

A ordem de eliminação dos vértices determina a ordem em que as linhas/colunas da matriz A serão reordenadas.

A nova ordem das linhas/colunas de A será $z = [10, 1, 6, 4, 5, 7, 8, 2, 3, 9]$ como apresentado na Tabela 4.5, em que as linhas/colunas mais esparsas estão nas primeiras posições e mais densas nas últimas posições da matriz.

Tabela 4.5 - Matriz *A* reordenada por MG

	10	1	6	4	5	7	8	2	3	9
10	10000	0	0	0	0	0	0	0	0	0
1	0	1000	0	14	0	0	0	0	0	0
6	0	0	6000	0	0	0	0	0	36	0
4	0	14	0	4000	0	0	48	24	0	0
5	0	0	0	0	5000	0	0	25	35	0
7	0	0	0	0	0	7000	78	0	37	79
8	0	0	0	48	0	78	8000	28	38	0
2	0	0	0	24	25	0	28	2000	23	29
3	0	0	36	0	35	37	38	23	3000	39
9	0	0	0	0	0	79	0	29	39	9000

Quando a fatoração de Cholesky é aplicada à matriz reordenada por MG o fator obtido possui menos elementos não nulos. Neste caso específico, a reordenação pelo MG não criou novas arestas sendo que novos elementos não nulos não foram adicionados ao fator; foi obtida uma reordenação ótima.

Tabela 4.6 - Fator L^t obtido após a fatoração por Cholesky da matriz *A* reordenada por MG

1.000.000	0	0	0	0	0	0	0	0	0
0	316.228	0	0,4427	0	0	0	0	0	0
0	0	774.597	0	0	0	0	0	0,4648	0
0	0	0	632.440	0	0	0,7590	0,3795	0	0
0	0	0	0	707.107	0	0	0,3536	0,4950	0
0	0	0	0	0	836.660	0,9323	0	0,4422	0,9442
0	0	0	0	0	0	894.346	0,3099	0,4203	-0,0098
0	0	0	0	0	0	0	447.173	0,5075	0,6486
0	0	0	0	0	0	0	0	547.623	0,6986
0	0	0	0	0	0	0	0	0	948.588

O custo computacional da implementação do MG é alto, e com isso, seu processamento pode se tornar lento. Diversas variações do MG foram desenvolvidas

com o intuito de reduzir o tempo de execução. Uma das variações mais utilizada é a *approximate minimum degree* (AMD) descrita em (Amestoy, et al., 1996).

4.3. Nested Dissection

Esse método consiste em representar a matriz A como um grafo G, buscando em G um conjunto de nós separadores S. Esse conjunto S irá tornar o grafo G bipartido.

Comumente o conjunto de vértices que compõem S será formado por linhas/colunas mais densas, dividindo o grafo em dois subgrafos distintos (George, 1973). Esses dois subgrafos podem ser nomeados C^1 e C^2.

Em seguida o método realiza a permutação das linhas/colunas da matriz de maneira que os vértices de C^1 e C^2 sejam alocados nas primeiras posições e as linhas/colunas de S sejam alocados nas últimas posições da matriz.

O processo é aplicado recursivamente em C^1e C^2 e por último em S. A maior dificuldade desse método de reordenação é encontrar os nós separadores para fazer a bisseção do grafo.

No ND pode-se adotar como critério de escolha do vértice inicial a estratégia de separação por vértices ou por arestas. A separação por vértices priorizar gerar o mínimo de vértices possível nas partições dos subgrafos e a separação por arestas busca gerar a menor quantidade de arestas em cada partição (Rothberg & Hendrickson, 1998).

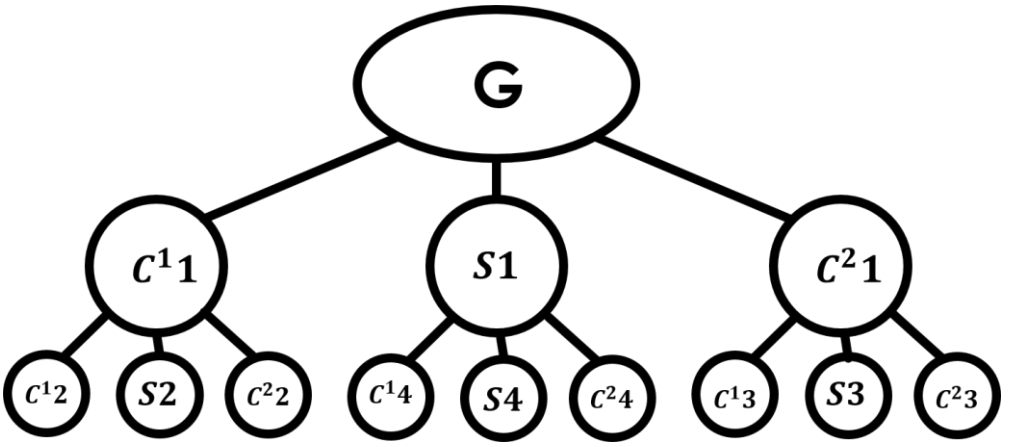

Figura 4.5 - Ilustração da reordenação ND

O ND é um algoritmo que usa a estratégia dividir e conquistar e segue as seguintes características de algoritmos que adotam essa estratégia, como descritas em (Cormen, 2013):

1. Divide o problema em vários subproblemas menores; O ND divide o grafo em C^1, C^2 e S.
2. Conquista os subproblemas resolvendo-os recursivamente. Se eles não forem suficientemente pequenos, resolve os subproblemas como casos-bases; Cada subgrafo é subdividido recursivamente novamente em C^1, C^2 e S até não haver mais a possibilidade de particionamento.
3. Combina as soluções para os subproblemas na solução para o problema original; Os subgrafos em C^1, C^2 e S são agrupados novamente compondo o grafo da matriz reordenada.

Procedimento 4.2: Nested Dissection

Entrada: Grafo de $\boldsymbol{A}$

Saída: Grafo $\boldsymbol{A'}$

Enquanto for possível particionar $\boldsymbol{A > 1}$ **faça**

1. Encontrar conjunto vértices de separadores $\boldsymbol{S}$ do grafo $\boldsymbol{A}$
2. Remover vértices $\boldsymbol{S}$ particionando grafo em $\boldsymbol{S}$, $\boldsymbol{C^1}$ e $\boldsymbol{C^2}$
3. Aplicar *Nested Dissection* recursivamente a $\boldsymbol{S}$, $\boldsymbol{C^1}$ e $\boldsymbol{C^2}$

Fim Enquanto

Compor $\boldsymbol{A'}$ re-rotular vértices de acordo com $\mathbf{C^1}$ e $\boldsymbol{C^2}$ e por último $\boldsymbol{S}$
Retorna $\boldsymbol{A'}$

Fim.

Esse trabalho fez uso da biblioteca METIS (Karypis & Kumar, 1999), em sua versão 5.0 e todas as opções em sua configuração padrão, com estratégia de escolha do nó separador por vértice. A biblioteca METIS é desenvolvida em linguagem C, com código fonte disponível para consulta e alterações.

Ao aplicar a reordenação ND na matriz *A* da Tabela 4.3 foi obtida a seguinte matriz reordenada:

Tabela 4.7 - Matriz reordenada com *Nested Dissection*

	7	9	4	6	3	2	1	8	10	5
7	7000	79	0	0	37	0	0	78	0	0
9	79	9000	0	0	39	29	0	0	0	0
4	0	0	4000	0	0	24	14	48	0	0
6	0	0	0	6000	36	0	0	0	0	0
3	37	39	0	36	3000	23	0	38	0	35
2	0	29	24	0	23	2000	0	28	0	25
1	0	0	14	0	0	0	1000	0	0	0
8	78	0	48	0	38	28	0	8000	0	0
10	0	0	0	0	0	0	0	0	10000	0
5	0	0	0	0	35	25	0	0	0	5000

A Tabela 4.8 apresenta o fator obtido na Fatoração de Cholesky da matriz reordenada por ND.

Tabela 4.8 - Fatoração de Cholesky após reordenação por *Nested Dissection*

836.660	0.9442	0	0	0.4422	0	0	0.9323	0	0
0	948.636	0	0	0.4067	0.3057	0	-0.0093	0	0
0	0	632.456	0	0	0.3795	0.2214	0.7589	0	0
0	0	0	774.597	0.4648	0	0	0	0	0
0	0	0	0	547.670	0.4177	0	0.6864	0	0.6391
0	0	0	0	0	447.168	-0.0019	0.6134	0	0.5531
0	0	0	0	0	0	316.220	-0.0053	0	0.0000
0	0	0	0	0	0	0	894.299	0	-0.0087
0	0	0	0	0	0	0	0	1.000.000	0
0	0	0	0	0	0	0	0	0	707.056

A matriz original apresentava 18 elementos nulos em sua triangular superior. Após a fatoração de Cholesky da matriz reordenada por ND, o fator obtido teve um aumento de 8 elementos no preenchimento com relação à matriz original.

5. EXPERIMENTOS NUMÉRICOS

Esta seção descreve experimentos que investigam o impacto do método ND na solução dos sistemas de métodos de pontos interiores (Equação 5.1),

$$(A\theta A^t)\Delta y = r, \tag{5.1}$$

pelo GCP com o precondicionador híbrido.

A investigação foi conduzida por meio da comparação com o método MG na solução dos sistemas da Equação (5.1), nas mesmas condições. Para isto, as implementações dos dois métodos de reordenação foram introduzidas no código PCx-Modificado (Oliveira & Sorensen, 2005), (Bocanegra, 2005), (Velazco, et al., 2010).

5.1. Especificações da Implementação

O código PCx, na sua versão original, foi desenvolvido por (Czyzyk, et al., 1999) no Optimization Technology Center do Argonne National Laboratory e da Universidade Northwestern nos Estados Unidos. Neste código, problemas de programação linear em formato MPS (formato de arquivo padrão de mercado para problemas de programação matemática) são resolvidos utilizando o método preditor-corretor com múltiplas correções (Gondzio, 1996). Os sistemas lineares para o cálculo das direções são resolvidos por fatoração de Cholesky e é utilizada a reordenação por MG. O código PCx é implementado nas linguagens C e Fortran.

No PCx-Modificado, as múltiplas correções e o método de reordenação foram desativados e os sistemas são resolvidos utilizando uma abordagem iterativa pelo GCP. O precondicionador utilizado é o precondicionador híbrido (Bocanegra, 2005) com as modificações descritas em (Velazco et al., 2010).

Em (Silva, 2015), testes com três métodos de reordenação, *Reverse Cuthill-McKee*, MG e *Sloan*, indicaram a redução no número de iterações do GCP. O melhor desempenho foi obtido com o método de reordenação MG e tal método foi reativado no PCx-Modificado.

Para a comparação do desempenho do ND com o MG foram utilizadas duas versões do PCx-Modificado descritas a seguir:

- PCx-Modificado com MG (PCx-MG): Foi utilizada a versão do PCx-Modificada com o MG reativado apresentada em (Silva, 2015);
- PCx-Modificado com ND (PCx-ND): Foi utilizada a versão do PCx-Modificada sem o MG. O ND utilizado foi o disponibilizado junto à biblioteca Metis desenvolvida por (Karypis & Kumar, 1999) na linguagem de programação C.

Os experimentos foram realizados em um processador Intel core I3 com 4GB de memória e sistema operacional Linux Ubuntu 16.04.

5.2. Problemas teste

O desempenho das configurações foi avaliado por testes numéricos em 63 problemas, sendo o menor dos problemas com 43 linhas de restrições e o maior com 152.289 linhas de restrições; tais problemas podem ser considerados de pequeno, médio e grande porte. Os problemas pertencem às bibliotecas de domínio público: PDS (Carolan, et al., 1990), MESZAROS, MNETGEN, NETLIB e QAP (Burkard, et al., 1991).

As descrições dos problemas são apresentadas na Tabela 5.1, por meio da categorização de cada um deles: número de ordem que o identifica a (Ordem), nome do problema (Nome), densidade em porcentagem, quantidade de elementos não nulos de $A\Theta A^t$ (NZAAT), dimensão da matriz $A\Theta A^t$ (m) e biblioteca de origem. Os problemas são organizados considerando a esparsidade da matriz dos coeficientes da Equação 5.1, $A\Theta A^t$; ordenados por ordem crescente das respectivas esparsidades. Note que mesmo os problemas menos esparsos ainda possuem densidade menor do que 40%.

Tabela 5.1 – Caracterização dos Problemas Teste

Ordem	Problema	Densidade (%)	NZAAT	*m*	*Biblioteca*
1	pds-100	0,0034	787355	152289	PDS
2	pds-90	0,0038	740086	139741	PDS
3	pds-80	0,0043	676093	126109	PDS
4	ken-18	0,0047	289462	78538	KENNINGTON
5	512-256-2	0,0064	705576	104931	MNETGEN
6	512-256-1	0,0066	713835	104124	MNETGEN
7	ken-13	0,0155	77459	22365	KENNINGTON
8	256-256-9	0,0158	674759	65280	MNETGEN

9	256-256-8	0,0162	693445	65328	MNETGEN
10	512-128-11	0,0221	1037581	68462	MNETGEN
11	256-256-10	0,0223	989269	66606	MNETGEN
12	256-256-12	0,0224	988399	66456	MNETGEN
13	256-256-11	0,0226	983752	66024	MNETGEN
14	nemsemm2	0,1873	38359	4526	MESZAROS
15	ste36a	0,2041	1564487	27683	QAPLIB
16	cre-a	0,2410	21527	2989	KENNINGTON
17	stocfor2	0,3032	11888	1980	NETLIB
18	pcb3000	0,3076	45639	3852	MESZAROS
19	cre-c	0,3138	17625	2370	KENNINGTON
20	chr25a	0,3755	249324	8149	QAPLIB
21	progas	0,6262	12449	1410	MESZAROS
22	scr20	0,6463	166709	5079	QAPLIB
23	orna1	0,6570	5111	882	MESZAROS
24	orna2	0,6570	5111	882	MESZAROS
25	orna7	0,6570	5111	882	MESZAROS
26	rou20	0,6586	356689	7359	QAPLIB
27	ganges	0,6914	8565	1113	NETLIB
28	nug12	0,7239	56508	2794	QAPLIB
29	els19	0,7284	137825	4350	QAPLIB
30	qap12	0,7708	60174	2794	NETLIB
31	nemscem	0,7919	1817	479	MESZAROS
32	pcb1000	0,8452	17211	1427	MESZAROS
33	nemspmm1	1,0153	50356	2227	MESZAROS
34	scfxm3	1,0272	8600	915	NETLIB
35	nesm	1,1052	4727	654	NETLIB
36	nemspmm2	1,1324	49040	2081	MESZAROS
37	scr15	1,1824	59009	2234	QAPLIB
38	bnl1	1,3026	4847	610	NETLIB
39	ship12l	1,4722	5478	610	NETLIB
40	scfxm2	1,5391	5727	610	NETLIB
41	qap08	1,6461	9063	742	NETLIB
42	nug08	1,6841	9272	742	QAPLIB
43	maros	1,8745	8042	655	NETLIB
44	25fv47	1,8749	11642	788	NETLIB
45	nug07	2,1894	4919	474	QAPLIB
46	etamacro	2,2339	2492	334	NETLIB
47	degen3	2,2919	51637	1501	NETLIB
48	model4	2,6386	38887	1214	MESZAROS

49	nug06	3,0587	2398	280	QAPLIB
50	ship04l	3,0634	2612	292	NETLIB
51	scfxm1	3,0680	2854	305	NETLIB
52	boeing1	3,8435	4211	331	NETLIB
53	capri	3,9290	2282	241	NETLIB
54	e226	6,4432	2526	198	NETLIB
55	recipe	6,7627	277	64	NETLIB
56	share1b	7,7089	967	112	NETLIB
57	boeing2	8,7050	1382	126	NETLIB
58	blend	14,2630	719	71	NETLIB
59	forplan	18,0042	2636	121	NETLIB
60	t0331-4l	22,4926	99169	664	MESZAROS
61	kb2	24,0671	445	43	NETLIB
62	nw14	36,4984	1945	73	MESZAROS
63	israel	37,0822	11227	174	NETLIB

5.3. Apresentação e discussão dos resultados numéricos

A comparação dos resultados foi realizada levando em consideração valores obtidos com relação aos parâmetros identificados a seguir:

1. Tempo de reordenação (Seção 5.3.1);
2. Total de iterações do GCP (Seção 5.3.2);
3. Iterações e Tempo total do método de pontos interiores (Seção 5.3.3);
4. Preenchimento do FCC depois do primeiro aumento do parâmetro η (Seção 5.3.4);
5. Iteração de troca de precondicionadores (Seção 5.3.5);

Os resultados obtidos para cada um dos 63 problemas, considerando cada um dos 5 parâmetros, são mostrados nas seções que seguem.

Por meio da ferramenta específica perfil de desempenho (Dolan & Moré, 2002) foi medido o desempenho de cada método de reordenação nos 63 problemas levando em consideração os parâmetros: iterações do GCP e tempo total de solução do método de pontos interiores.

O conceito perfil de desempenho é baseado na proposta descrita no artigo de (Dolan & Moré, 2002) e vem tendo um crescente aumento de sua utilização para

comparação de implementações científicas (Munari, 2009). O perfil de desempenho tem como objetivo evidenciar o desempenho dos métodos empregados, por meio do exame dos valores das métricas de avaliação.

Considere um conjunto P com n_p problemas e um conjunto S com n_s implementações. Para cada problema $p \in P$ e implementação $s \in S$ existe $t_{p,s}$ baseado no parâmetro de comparação definido.

A Equação 5.2 calcula a razão de desempenho $r_{p,s}$. A razão $r_{p,s}$ mostra o comportamento de uma implementação s em apenas um problema p. Um software s, terá a comparação de seu desempenho na resolução de um problema p em relação ao desempenho da melhor implementação.

$$r_{p,s} = \frac{t_{p,s}}{\min\{\ t_{p,s} : \forall\ s \in S\}}. \tag{5.2}$$

O perfil de desempenho $p_s(\tau)$ é particular a cada implementação e é dado pela Equação 5.3.

$$p_s(\tau) = \frac{1}{n_p}\left|\{p \in P : r_{p,s} \leq \tau\}\right|. \tag{5.3}$$

O perfil de desempenho $p_s(\tau)$ calcula a fração de problemas resolvidos pela implementação com um desempenho menor ou igual a um fator de desempenho τ.

5.3.1. Tempo de reordenação

A Tabela 5.2 mostra o tempo, em segundos, consumido na reordenação da matriz de restrições A utilizando os dois métodos de reordenação, ND e MG, respectivamente. Tempos inferiores àquele (0,004s) que pode ser medido pela função utilizada na linguagem de programação C, foram considerados 0. Na tabela, os melhores resultados são mostrados em negrito.

Tabela 5.2 – Comparativo de tempo de reordenação entre ND e MG

Ordem	Problema	ND (s)	MD(s)

1	pds-100	**3,124**	323,412
2	pds-90	**2,936**	314,584
3	pds-80	**2,692**	409,696
4	ken-18	0,708	**0,664**
5	512-256-2	**1,480**	694,496
6	512-256-1	**1,436**	602,708
7	ken-13	0,200	**0,088**
8	256-256-9	**0**	556,572
9	256-256-8	**0**	606,088
10	512-128-11	**0**	743,980
11	256-256-10	**0,004**	786,508
12	256-256-12	**0**	683,036
13	256-256-11	**0**	863,108
14	nemsemm2	**0**	0,012
15	ste36a	**1,400**	29,240
16	cre-a	0,024	**0,012**
17	stocfor2	0,016	**0**
18	pcb3000	**0**	0,024
19	cre-c	0,02	**0,016**
20	chr25a	**0,268**	0,452
21	progas	0,016	**0**
22	scr20	**0,124**	0,648
23	orna1	0,004	**0**
24	orna2	0,004	**0**
25	orna7	0,004	**0**
26	rou20	0,240	**0,220**
27	ganges	**0**	0,004
28	nug12	**0,052**	0,416
29	els19	**0,096**	0,720
30	qap12	**0,048**	0,428
31	nemscem	0,004	0,004
32	pcb1000	**0**	0,004
33	nemspmm1	**0**	0,032
34	scfxm3	**0**	0,004
35	nesm	0,004	0,004
36	nemspmm2	**0**	0,016
37	scr15	**0,04**	0,072
38	bnl1	**0**	0,004
39	ship12l	0,004	0,004
40	scfxm2	0	0
41	qap08	**0,012**	0,024
42	nug08	**0,008**	0,016
43	maros	**0**	0,004

44	25fv47	**0**	0,008
45	nug07	0,004	0,004
46	etamacro	0	0
47	degen3	**0**	0,272
48	model4	**0**	0,016
49	nug06	0,004	0,004
50	ship04l	0,004	0,004
51	scfxm1	0	0
52	boeing1	0,004	**0**
53	capri	0	0
54	e226	0	0
55	recipe	0	0
56	share1b	0,004	**0**
57	boeing2	0,004	**0**
58	blend	0	0
59	forplan	0	0
60	t0331-4l	0,032	**0**
61	kb2	0	0
62	nw14	0,004	**0**
63	israel	0	0
TOTAL		**15,028**	**6617,628**

A heurística ND comparada com a MG consumiu menos tempo para reordenação da matriz em 33 problemas, igualou em 15 problemas e consumiu mais tempo em 15 problemas.

O tempo total consumido pelo ND na reordenação, considerando todos os problemas, foi de 15,028 segundos, enquanto que o MG necessitou 6617,628 segundos. Esta diferença é mais notória nos problemas com matrizes mais esparsas, como mostra a Tabela 5.2 nos primeiros 15 problemas. À medida que aumenta a densidade da matriz dos coeficientes $A\Theta A^t$, o desempenho do MG melhora, com resultados superiores comparados com o ND. O MG é melhor que o ND em 15 problemas. Porém, o tempo total consumido na reordenação destes 15 problemas pelo MG foi de 1s e pelo ND 1,208s; isto é, o MG foi apenas 0,208 segundos mais rápido.

Podemos destacar os problemas PDS-80(3), com 126109 linhas e preenchimento 0,0042%, onde método de reordenação ND consumiu 2,692 segundos enquanto o MG precisou 409,969 segundos para reordenar o mesmo problema. Já na biblioteca

MNETGEN, temos o problema 256-256-11, com 983752 linhas e preenchimento 0,022%, onde o MG consumiu 863,108 segundos e não foi possível efetuar a medição precisa do tempo de reordenação do ND, pois a função utilizada na linguagem de programação C não mede tempo inferior a 0,004 segundos.

5.3.2. Total de iterações do GCP

A Tabela 5.3 mostra o número total de iterações do GCP para solucionar os dois sistemas lineares durante o processo de otimização, quando utilizado o PCx-ND e o PCx-MG.

Tabela 5.3 - Total de iterações do GCP
(*) Método não convergiu

Ordem	Problema	GCP - ND	GCP - MG
1	pds-100	**47004**	49285
2	pds-90	**40420**	42984
3	pds-80	**36905**	36944
4	ken-18	**27978**	36458
5	512-256-2	*	**23674**
6	512-256-1	23766	**16021**
7	ken-13	20461	**16019**
8	256-256-9	51242	**40407**
9	256-256-8	41427	**31550**
10	512-128-11	71269	**64961**
11	256-256-10	**73987**	75549
12	256-256-12	61239	**52736**
13	256-256-11	63725	**59831**
14	nemsemm2	7103	**5208**
15	ste36a	49099	**49010**
16	cre-a	5017	**4484**
17	stocfor2	6749	**4652**
18	pcb3000	15062	**14483**
19	cre-c	5636	**5281**
20	chr25a	**16089**	18603
21	Progas	5146	**4632**
22	scr20	17705	**16819**
23	orna1	2746	**2488**
24	orna2	2355	**2151**
25	orna7	3416	**2790**
26	rou20	16056	**14910**
27	Ganges	**2946**	3115

28	nug12	22157	**19307**
29	els19	**24941**	25248
30	qap12	19741	**17626**
31	Nemscem	1916	**1526**
32	pcb1000	7607	**6487**
33	nemspmm1	*	**28697**
34	scfxm3	**2412**	3638
35	Nesm	11712	**7818**
36	nemspmm2	23491	**22816**
37	scr15	14797	**13837**
38	bnl1	16179	**4549**
39	ship12l	**235**	256
40	scfxm2	1387	**482**
41	qap08	**3856**	4030
42	nug08	**2996**	3237
43	Maros	5305	**3767**
44	25fv47	5529	**5488**
45	nug07	**1979**	*
46	etamacro	1131	**983**
47	degen3	9497	**9364**
48	model4	20100	**12311**
49	nug06	651	**567**
50	ship04l	105	**104**
51	scfxm1	1051	**398**
52	boeing1	**908**	1034
53	Capri	823	**577**
54	e226	1375	**1014**
55	Recipe	60	60
56	share1b	**41**	212
57	boeing2	560	**543**
58	Blend	**279**	348
59	Forplan	**736**	*
60	t0331-4l	10171	**9402**
61	kb2	**199**	218
62	nw14	986	**735**
63	Israel	1128	**642**
TOTAL		**930589**	**902366**

O GCP teve um desempenho melhor quando usado o PCx-MG. Com o PCx-MG foram necessárias 902366 iterações do GCP (14792 iterações em média por problema) e com o PCx-ND foram necessárias 930589 iterações (15256 iterações em média por

problema). O PCx-ND realizou 3% a mais de iterações; apesar da diferença, na média, não ter sido tão significativa, o desempenho do PCx-ND foi pior em 45 problemas.

Na Figura 5.1 é apresentado o perfil de desempenho em relação à quantidade de iterações do GCP considerando as duas versões do código PCx-Modificado utilizadas neste trabalho. O PCx-MG teve melhor desempenho, $p_s(\tau)$, para valores pequenos do fator de desempenho, τ. A Curva de PCx-MG domina a curva de PCx-ND.

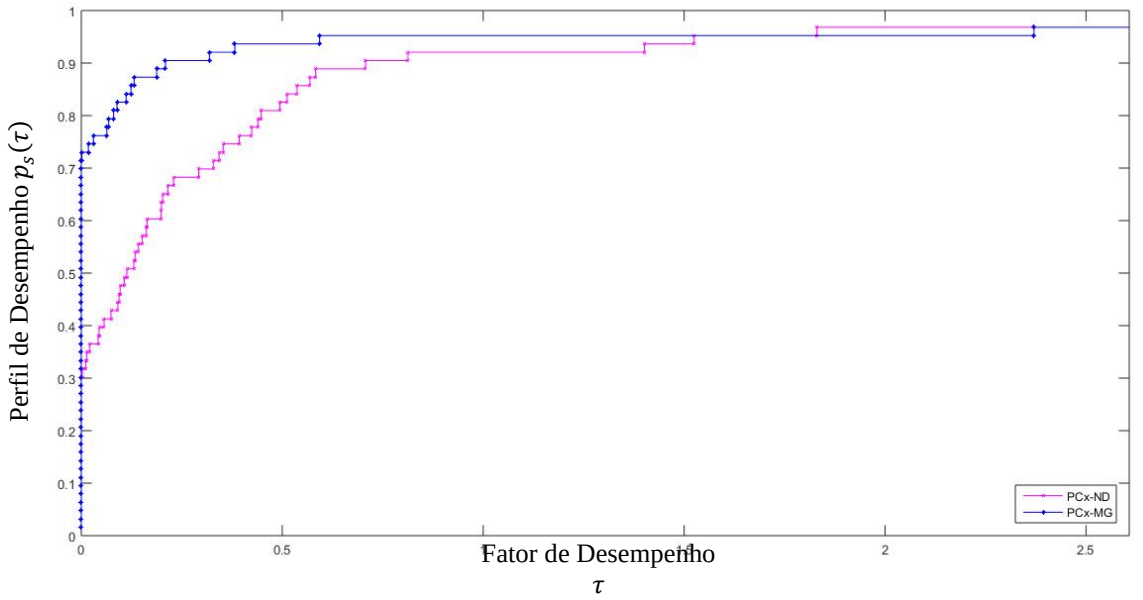

Figura 5.1 - Perfil de desempenho do total de iterações do GCP

5.3.3. Iterações e tempo total do método de pontos interiores

O desempenho do GCP tem muita influência no desempenho do método de pontos interiores. A Tabela 5.4 mostra o número de iterações do método de ponto interiores e tempo total de otimização para cada uma das versões do PCx-Modificado.

Tabela 5.4 - Total de iterações dos métodos de ponto interiores e tempo total de solução.
(*) Método não convergiu

Ordem	Problema	Iterações MPI- ND	Tempo MPI- ND	Iterações MPI- MG	Tempo MPI- MG
1	pds-100	**85**	**2992,36**	88	4428,58
2	pds-90	**80**	**2741,80**	81	3163,68
3	pds-80	83	**2017,40**	83	2692,81
4	ken-18	**39**	**2089,36**	41	2440,82
5	512-256-2	*	*	**49**	**3387,92**
6	512-256-1	**53**	**1763,04**	54	3124,90
7	ken-13	**29**	267,29	30	**194,90**
8	256-256-9	51	5753,34	**40**	**2772,14**
9	256-256-8	50	5124,75	**48**	**2940,90**
10	512-128-11	51	9384,48	**49**	**8519,54**
11	256-256-10	**61**	**4790,90**	64	6898,15
12	256-256-12	**49**	7975,22	51	**7335,32**
13	256-256-11	**57**	**7344,32**	61	7481,93
14	nemsemm2	44	14,55	**43**	**9,70**
15	ste36a	38	**9890,73**	38	10244,54
16	cre-a	**27**	3,24	28	**3,22**
17	stocfor2	21	2,18	21	**1,58**
18	pcb3000	28	64,70	28	**38,34**
19	cre-c	29	3,88	**28**	**3,62**
20	chr25a	28	**87,64**	28	160,54
21	progas	16	1,76	16	**1,57**
22	scr20	21	102,27	21	**99,70**
23	orna1	25	0,42	25	**0,39**
24	orna2	19	0,36	19	**0,35**
25	orna7	20	0,49	**19**	**0,42**
26	rou20	24	**1154,68**	24	1169,06
27	ganges	19	**1,36**	19	1,69
28	nug12	20	147,88	20	**134,62**
29	els19	**18**	87,77	31	**78,76**
30	qap12	20	113,93	20	**110,23**
31	nemscem	22	0,16	22	**0,13**
32	pcb1000	25	9,19	25	**5,40**
33	nemspmm1	*	*	**54**	**34,51**
34	scfxm3	**20**	**0,74**	25	0,86
35	nesm	**32**	3,10	33	**2,38**
36	nemspmm2	45	28,04	45	**25,73**
37	scr15	24	16,55	24	**15,74**
38	bnl1	42	2,22	42	**0,73**
39	ship12l	17	**0,09**	17	0,10
40	scfxm2	20	0,34	20	**0,14**

41	qap08	10	**1,46**	10	1,47
42	nug08	10	1,22	**9**	**1,19**
43	maros	20	1,11	20	**0,66**
44	25fv47	26	1,69	26	**1,53**
45	nug07	**11**	**0,42**	*	*
46	etamacro	30	0,21	**27**	0,21
47	degen3	16	11,47	16	**7,92**
48	model4	40	14,85	**39**	**9,08**
49	nug06	6	0,10	6	0,10
50	ship04l	18	0,03	18	0,03
51	scfxm1	17	0,14	17	**0,05**
52	boeing1	20	**0,11**	20	0,13
53	capri	20	0,08	20	**0,06**
54	e226	18	0,12	18	**0,07**
55	recipe	12	0,00	12	0,00
56	share1b	19	**0,01**	19	0,02
57	boeing2	14	0,03	14	0,03
58	blend	10	0,01	10	0,01
59	forplan	**23**	**0,11**	*	*
60	t0331-4l	42	252,03	42	**244,80**
61	kb2	13	0,00	13	0,00
62	nw14	48	16,14	**47**	**13,14**
63	israel	21	0,54	21	**0,31**
TOTAL		**1816**	**64284,41**	**1898**	**67806,45**

O método de pontos interiores obteve uma solução ótima quando o PCx-ND foi utilizado em 61 problemas. O PCx-MG teve o mesmo desempenho, mas em problemas diferentes. O PCx-ND diminuiu o número de iterações do preditor-corretor em 14 problemas e igualou em 37 problemas enquanto o PCx-MG diminuiu em 12 problemas. Porém, o PCx-MG conseguiu diminuir o tempo total de resolução em 38 problemas com uma redução de 23% do tempo e o PCx-ND, em 18 problemas com uma redução de 16% do tempo. Desconsiderando em ambas as versões os problemas que não foram resolvidos, no tempo total de solução o PCx-ND (64283,88 segundos) teve um melhor comportamento se comparado com PCx-MG (64384,02 segundos). Esta diferença de tempo se deve à da diferença de tempo das reordenações.

A redução do tempo de solução em alguns problemas foi obtida pela redução do tempo de reordenação com o ND. O problema 256-256-11(13) consumiu 7344,32

segundos com o PCx-ND e 7481,93 segundos com o PCx-MD, sendo que a diferença de tempo de reordenação foi de 863,108 segundos.

Na Figura 5.2, é apresentado o perfil de desempenho considerando o tempo para solução do problema com o método de pontos interiores. O gráfico da Figura 5.2 mostra que o PCx-MG conseguiu resolver mais problemas em menos tempo do que PCx-ND.

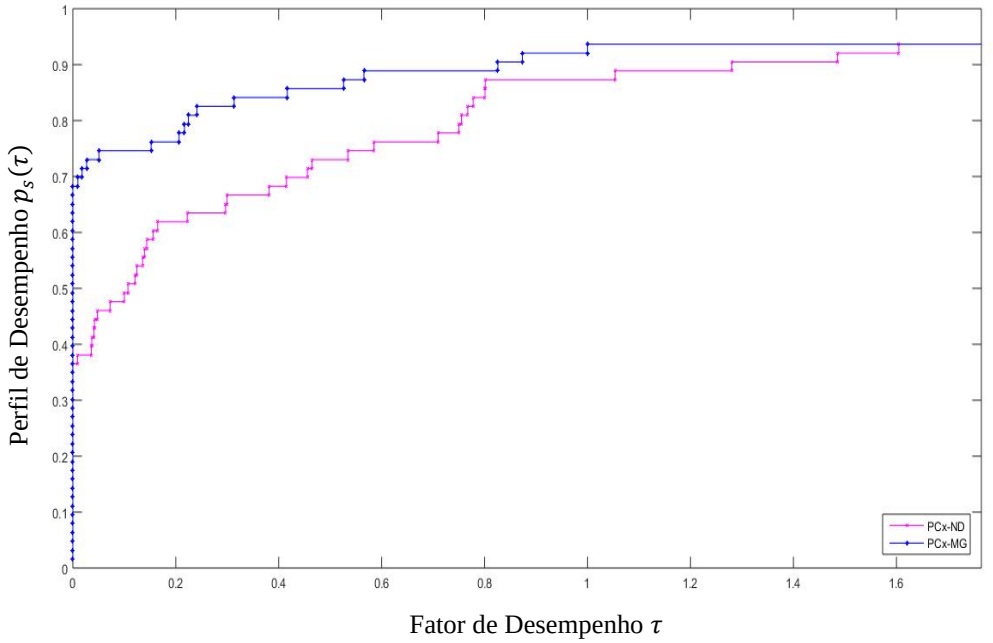

Figura 5.2 - Perfil de desempenho do tempo total para solução dos problemas

5.3.4. Preenchimento do FCC depois do primeiro aumento do parâmetro η

Quando os sistemas do método de pontos interiores representados na Equação 5.1 são resolvidos usando o método direto pela fatoração de Cholesky, a reordenação da matriz dos coeficientes $A\theta A^t$ reduz o preenchimento do fator obtido na fatoração. Quando os sistemas (Equação 5.1) são resolvidos pelo GCP com um precondicionador obtido a partir de uma fatoração incompleta, a reordenação ajuda na convergência do GCP (Silva, 2015). A reordenação da matriz permite criar um precondicionador FCC em que a norma do resíduo (Equação 3.6) será menor.

A Tabela 5.5 mostra a quantidade de elementos não nulos do FCC depois do primeiro aumento do η, usando PCx-ND (NZFCC-ND) e PCx-MG (NZFCC-MG), bem

como identifica a iteração em que ocorreu o primeiro aumento e o η inicial (η_0) para cada problema. Na Tabela 5.5 o símbolo "-" significa que não houve aumento no valor de η.

O parâmetro η determina o preenchimento a mais que será permitido por coluna no FCC. O valor $\eta_0 \in \mathbb{Z}$ é calculado a partir da Equação 5.3:

$$\eta_0 = \begin{cases} -30 & \frac{|A\theta A^t|}{m} \geq 30 \\ -\frac{|A\theta A^t|}{m} & se\ 10 \leq \frac{|A\theta A^t|}{m} < 30 \\ \frac{|A\theta A^t|}{m} & caso\ contrário \end{cases} \quad (5.\ 3)$$

em que $|A\theta A^t|$ representa a quantidade de elementos não nulos da matriz $A\theta A^t$ e m é a dimensão da matriz. O termo $\frac{|A\theta A^t|}{m}$, na Equação 5.3, calcula a media dos elementos não nulos por coluna. Se o preenchimento por coluna for maior que 30 então as colunas serão muito cheias e o valor de η_0 será negativo; com isto o precondicionador FCC inicialmente terá menos elementos que $A\theta A^t$ por colunas. Se a media dos elementos for menor que 30, isto é, a matriz for esparsa então $0 \leq \eta_0 < 10$ pelo que será permitido um preenchimento maior por coluna de até *10* elementos.

Quando o FCC perde eficiência, isto é, as iterações do GCP são maiores que $\frac{m}{6}$ então é permitido um maior preenchimento do FCC e o valor de η será atualizado como: $\eta = \eta + 10$.

A Tabela 5.5 mostra que depois do primeiro aumento do valor de η quando usado o PCx-ND o preenchimento do FCC foi maior em 36 problemas pelo que a reordenação da matriz não foi melhor nesses casos. Um maior preenchimento do FCC melhora a convergência do GCP, mas encarece a construção do mesmo pelo uso da memória. Para valores "razoáveis" de η, o cálculo do FCC é barato e requer menos memória que o precondicionador separador.

Tabela 5.5 - Elementos não nulos do FCC depois do primeiro aumento do η

Ordem	Problema	η Inicial	NZFCC - ND	Iteração Primeiro η - ND	NZFCC – MG	Iteração Primeiro η - MG
1	pds-100	5	-	-	-	-
2	pds-90	5	-	-	-	-
3	pds-80	5	-	-	-	-
4	ken-18	3	630285	29	**590095**	**30**
5	512-256-2	6	1309767	**44**	**1258884**	43
6	512-256-1	6	1336877	46	**1285375**	**47**
7	ken-13	3	166743	25	**158215**	25
8	256-256-9	-10	674759	30	674759	**33**
9	256-256-8	-10	693445	**37**	693445	33
10	512-128-11	-15	672910	36	**659521**	**38**
11	256-256-10	-14	696787	42	**688593**	42
12	256-256-12	-14	696643	42	**688195**	42
13	256-256-11	-14	693356	42	**684666**	**44**
14	nemsemm2	8	**78879**	39	-	**-**
15	ste36a	-30	**996835**	21	996885	21
16	cre-a	7	29906	22	**29743**	22
17	stocfor2	6	23254	12	**22389**	**13**
18	pcb3000	-11	**39107**	8	39792	8
19	cre-c	7	25607	22	**25488**	22
20	chr25a	-30	82145	9	**82088**	9
21	progas	8	24769	8	**24280**	8
22	scr20	-30	62633	7	**62632**	**9**
23	orna1	5	**6510**	14	7216	**15**
24	orna2	5	7893	12	**7216**	12
25	orna7	5	7893	**9**	**7216**	8
26	rou20	-30	205754	7	**205722**	7
27	ganges	7	21275	12	**17473**	12
28	nug12	-20	**27099**	1	27122	1
29	els19	-30	48750	8	**48717**	8
30	qap12	-21	**28060**	1	28120	1
31	nemscem	3	2798	18	**2535**	18
32	pcb1000	-12	**13585**	6	13671	6
33	nemspmm1	-22	**22578**	1	22606	1
34	scfxm3	9	19857	12	**13458**	**17**
35	nesm	7	11865	5	**11116**	**6**
36	nemspmm2	-23	**20924**	1	20962	1
37	scr15	-26	**22179**	6	22186	6
38	bnl1	7	12265	11	**8521**	**32**
39	ship12l	8	-	-	-	-
40	scfxm2	9	13277	10	**8858**	**16**

41	qap08	-12	7226	1	**7211**	1
42	nug08	-12	**7392**	1	7417	1
43	maros	-12	6429	2	**6411**	2
44	25fv47	-14	8110	6	**8095**	6
45	nug07	-10	4919	1	4919	1
46	etamacro	7	-	**-**	**5860**	20
47	degen3	-30	21155	**4**	**20940**	3
48	model4	-30	14041	**8**	**14024**	1
49	nug06	8	6206	2	**6114**	2
50	ship04l	8	-	-	-	-
51	scfxm1	9	6581	7	**4254**	**13**
52	boeing1	-12	3381	1	3381	1
53	capri	9	4185	13	**3724**	13
54	e226	-12	2024	1	**2019**	1
55	recipe	4	-	-	-	-
56	share1b	8	-	**-**	**1413**	1
57	boeing2	-10	1382	1	1382	1
58	blend	-10	719	1	719	1
59	forplan	-21	1241	1	**1239**	1
60	t0331-4l	-30	**85566**	11	85572	**13**
61	kb2	-10	445	1	445	1
62	nw14	-26	748	2	**742**	2
63	israel	-30	7661	1	**7659**	1

5.3.5. Troca de precondicionador

A Tabela 5.6 mostra a iteração de troca de precondicionadores para o PCx-ND (coluna ND) e PCx-MG (coluna MG). Como foi discutido no Capítulo 3, o precondicionamento na versão do PCx-Modificado é feita pelo precondicionador híbrido. Este precondicionador, nas iterações iniciais, usa o FCC. À medida que a matriz do sistema (Equação 5.1) fica mal condicionada, o precondicionador perde a eficiência, aumentando a quantidade de iterações do GCP. Quando a quantidade de iterações do GCP atinge um valor superior a $\frac{m}{6}$ para convergir e o preenchimento permitido no FCC atinge o seu máximo (η_{max}) é feita a troca de precondicionadores. O valor de η_{max} depende do problema e foi considerado $10 \leq \eta_{max} \leq 70$. Nas iterações seguintes, a matriz do sistema será precondicionada pelo precondicionador separador. Em alguns problemas não ocorre troca de precondicionadores e o FCC é utilizado durante todo o

processo de otimização. Na Tabela 5.6, o símbolo " – " indica que não houve troca de precondicionador e "*", que o método não convergiu.

Tabela 5.6 - Iteração de troca do precondicionador

Ordem	Problema	η_{max}	ND	MG
1	pds-100	10	-	-
2	pds-90	10	-	-
3	pds-80	10	-	-
4	ken-18	30	33	34
5	512-256-2	10	*	45
6	512-256-1	10	48	49
7	ken-13	30	29	29
8	256-256-9	10	38	38
9	256-256-8	10	40	39
10	512-128-11	10	43	42
11	256-256-10	10	52	52
12	256-256-12	10	46	49
13	256-256-11	10	52	52
14	nemsemm2	30	-	-
15	ste36a	70	33	33
16	cre-a	30	27	-
17	stocfor2	30	-	-
18	pcb3000	30	16	24
19	cre-c	30	-	-
20	chr25a	30	18	24
21	progas	30	-	-
22	scr20	30	17	17
23	orna1	30	-	-
24	orna2	30	-	-
25	orna7	30	-	-
26	rou20	30	14	14
27	ganges	30	16	16
28	nug12	80	14	14
29	els19	30	31	17
30	qap12	70	15	14
31	nemscem	10	20	20
32	pcb1000	30	12	18
33	nemspmm1	70	*	42
34	scfxm3	30	-	23
35	nesm	30	10	21
36	nemspmm2	30	43	42

37	scr15	30	14	14
38	bnl1	10	14	37
39	ship12l	10	-	-
40	scfxm2	20	-	-
41	qap08	30	7	7
42	nug08	10	5	6
43	maros	30	14	19
44	25fv47	30	-	-
45	nug07	10	5	*
46	etamacro	10	-	-
47	degen3	30	11	10
48	model4	30	29	-
49	nug06	10	4	4
50	ship04l	10	-	-
51	scfxm1	10	12	16
52	boeing1	10	-	-
53	capri	10	18	17
54	e226	10	13	-
55	recipe	10	-	-
56	share1b	10	-	-
57	boeing2	10	-	-
58	blend	10	-	-
59	forplan	10	23	*
60	t0331-4l	10	16	18
61	kb2	10	-	-
62	nw14	10	34	39
63	israel	10	17	-

Na Tabela 5.6 é possível notar que houve antecipação da troca de precondicionador quando usado o PCx-ND e, inclusive, houve troca em problemas onde não ocorria troca com o PCx-MG. A antecipação da troca de precondicionadores significa perda de eficiência no precondicionamento pelo FCC o que pode provocar perda de eficiência no precondicionador separador. O precondicionador separador foi construído para trabalhar com matrizes muito mal condicionadas e isto ocorre no final do processo iterativo dos métodos de pontos interiores.

Em 21 problemas houve antecipação da troca de precondicionadores quando utilizado o PCx-ND. Nos problemas ken18 (4), chr25a(20), els19(29), nug08(42) houve diminuição das iterações do GCP, como exemplo, nos problemas ken18 (4) e chr25a(20)

houve diminuição do tempo total do método preditor-corretor. Nos problemas, els19(29) e nug08(42) houve um aumento do tempo total do preditor-corretor porque o precondicionador separador precisou construir mais vezes a matriz B.

No problema scfxm3(34) não houve troca de precondicionadores quando o PCx-ND foi utilizado; adicionalmente, as iterações do GCP diminuíram e o tempo do preditor-corretor também.

No problema ste36a(15), a troca se manteve na mesma iteração quando as duas reordenações são utilizadas. O PCx-ND construiu um FCC menos cheio, aumento o número de iterações do GCP, porém terminou o processo de otimização em menos tempo computacional. Para este problema podemos concluir que a reordenação ND permitiu um melhor desempenho.

6. CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foi investigado o impacto da reordenação de matriz oriunda de métodos de pontos interiores pelo ND. Para investigação foram utilizadas duas versões do código PCx-Modificado: PCx-MG e PCx-ND. O código PCx-Modificado usa o método iterativo GCP para resolução de sistemas lineares com o precondicionador híbrido (Bocanegra, 2005) (Velazco et al., 2010).

Na versão PCx-MG, o MG é utilizado na reordenação de matrizes e na versão PCx-ND, o ND é utilizado. A investigação foi realizada através da comparação dos resultados obtidos em 63 problemas de bibliotecas de domínio público.

Com base nos resultados obtidos e apresentados no Capítulo 5, é possível identificar que o método de reordenação ND efetuou a reordenação da maioria dos problemas com menor consumo de tempo. A diferença de tempo de reordenação é mais acentuada em problemas de maior porte e mais esparsos, como pode ser observado nos problemas das bibliotecas PDS e MNETGEN.

Outro fator relevante em relação aos dois métodos de reordenação a ser notado é que dos 63 problemas testados, cada uma das duas versões do PCx-Modificado não foram capazes de chegar a uma solução ótima em apenas 2 dos problemas.

Porém o ND se mostrou menos eficiente com relação à redução do efeito de preenchimento durante a FCC. O PCx-ND antecipou a troca do precondicionador em 21 problemas. Foi também necessário um número maior de iterações do GCP para a solução de todos os problemas, quando o PCx-ND foi utilizado.

Com relação ao tempo total de execução dos 63 problemas abordados, a versão PCx-ND necessitou menos tempo para processamento total dos problemas, diferença essa dada pela diferença no tempo de reordenação entre as duas versões. Mesmo tendo mais iterações do GCP, o PCx-ND consumiu menos tempo para obtenção do resultado final.

Na maioria dos problemas a reordenação pelo ND levou a um pior desempenho do precondicionador FCC, gerando antecipação da troca de precondicionador, porém o seu custo computacional é menor, gerando redução no tempo total de processamento dos métodos de pontos interiores.

6.1. Trabalhos Futuros

Estes métodos de reordenação devem ser testados em problemas de maior porte e mais esparsos, e com isto confirmar quando deve ser usado cada um.

Um experimento necessário seria o de alterar a versão original do código PCx, em que os sistemas lineares são resolvidos pela fatoração completa de Cholesky, e realizar testes com o método de reordenação ND, para, em seguida comparar os resultados obtidos com a versão original que usa o MG. Este experimento tem por objetivo investigar se a matriz oriunda de métodos de pontos interiores reordenada por ND em relação ao MG provoca um maior preenchimento ao calcular a fatoração completa de Cholesky.

Outra abordagem para ser adotada em estudos futuros seria a utilização de uma reordenação hibrida entre o ND e MG, como feita em (Rothberg & Hendrickson, 1998), para verificação da efetividade em métodos de ponto interiores, mas quando usados métodos iterativos na solução dos sistemas.

7. BIBLIOGRAFIA

Amestoy, P. R., Davis, T. A. & Duff, I. S. (1996). An Approximate Minimum Degree Ordering Algorithm: SIAM Journal on Matrix Analysis and Applications. Vol. 17, nº4, pp. 886-905, DOI: 10.1137/S0895479894278952.

Bazaraa, M. S., Jarvis, J. J. & Sherali, H. D. (2010). Linear Programming and Network Flows (4. ed.). New Jersey: John Wiley & Sons, Inc., 764 p. ISBN 978-0-470-46272-0.

Benzi, M. (2002). Preconditioning Techniques for Large Linear Systems: A Survey. Journal of Computational Physics. Atlanta, 60 p. DOI: 10.1006/jcph.2002.7176.

Bocanegra, S. (2005). Algoritmos de Newton-Krylov precondicionados para métodos de pontos interiores. Programa de Doutorado em Ciência da Computação, Universidade Federal de Minas Gerais, 109 p.

Burkard, R. E., Karisch, S. E. & Rendl, F. (1991). QAPLIB - A Quadratic Assignment Problem. European Journal of Operations Research. Vol. 55, nº 1, pp. 115-119. DOI: 10.1016/0377-2217(91)90197-4.

Campos, F. F. & Birkett, N. R. C. (1998). An Efficient Solver for Multi-Right-Hand-Side Linear Systems Based on the CCCG (η) Method with Applications to Implicit Time-Dependent Partial Differential Equations. SIAM Journal on Scientific Computing. Vol. 19, nº 1, pp. 126-138. DOI: 10.1137/S106482759630382X.

Carmo, F. C. d. (2005). Análise da Influência de Algoritmos de Reordenação de Matrizes Esparsas no Desempenho do Método CCCG(η). Programa de Mestrado em Ciência da Computação, Universidade Federal de Minas Gerais, 151 p.

Carolan, W. Hill, J.E., Niemi, S. & Wichmann, S.J. (1990). An Empirical Evaluation of the KORBX® Algorithms for Military Airlift Applications. Operation Research. Vol. 38, nº 2, pp. 240-248. DOI: 10.1287/opre.38.2.240.

Cormen, T. H. (2013). Desmistificando Algoritmos. Rio de Janeiro: Elsevier, 2013. 200 p. ISBN: 978-85-352-7177-5.

Cuthill, E. & Mckee, J. (1969). Reducing the Bandwidth of Sparse Symmetric Matrices. Proceedings of the 24th National Conference ACM. pp. 157-172. DOI: 10.1145/800195.805928.

Czyzyk, J., Wagner, M. & Wright, S.J. (1999). PCx: an Interior-Point Code for Linear Programming. Optimization Methods and Software. Vol. 11&12, pp. 397-430. DOI: 10.1080/10556789908805757.

Davis, T. A., Rajamanickam, S. & Sid-Lakhdar, W. M. (2016). A survey of direct methods for sparse linear systems. Texas: Acta Numerica 25, pp. 383–566. Disponível em: <http://faculty.cse.tamu.edu/davis/publications_files/survey_tech_report.pdf>. Acesso em: 13 nov. 2016.

Dolan, E. D. & Moré, J. J. (2002). Benchmarking Optimization Software with Performance Profiles. Mathematical Programming, Vol. 91, nº 2, pp. 201-213. DOI: 10.1007/s101070100263.

George, A. (1973). Nested Dissection of a Regular Finite Element Mesh. SIAM Journal on Numerical Analysis. Vol. 10, nº 2, pp. 345-363. DOI: 10.1137/0710032.

George, A., Liu, J. & NG, E. (1994). Computer Solution of Sparse Linear Systems. 401 p. Disponível em: <http://web.engr.illinois.edu/~heath/courses/cs598mh/george _liu.pdf>. Acesso em: 28 abr. 2015.

Golub, G. H. & Van Loan, C. F. (2013). Matrix Computations (4th ed.). Baltimore: The John Hopkins University Press. 756 p. ISBN 13: 978-1-4214-0794-4.

Gomes, W. d. P. (2009). Programação de tripulantes de aeronaves no contexto brasileiro. Programa de Mestrado em Engenharia, Universidade de São Paulo, 111 p. DOI: 10.11606/D.3.2009.tde-21122009-170008.

Gondzio, J. (1996). Multiple Centrality Corrections in a Primal-Dual Method for Linear Programming. Computational Optimization and Applications. Vol. 6, nº 2, pp. 137-156. DOI: 10.1007/BF00249643.

Gondzio, J. (2012). Interior Point Methods 25 Years Later. European Journal of Operational Research. Vol. 218, nº 3, pp. 587-601. DOI 10.1016/j.ejor.2011.09.017.

Karmarkar, N. (1984). A New Polynomial-Time Algorithm for Linear Programming. STOC '84 Proceedings of the sixteenth annual ACM symposium on Theory of computing. pp. 302-311, ISBN: 0-89791-133-4, DOI: 10.1145/800057.808695.

Karypis, G. & Kumar, V. (1999). A Fast And High Quality Multilevel Scheme For Partitioning Irregular Graphs. SIAM Journal on Scientific Computing. Vol. 20, nº 1, pp 359-392. DOI: 10.1137/S1064827595287997.

Khachiyan, L. G. (1980). Polynomial algorithm in linear programming: U.S.S.R. Computational Mathematics and Mathematical Physics, Vol. 20, nº 1, pp. 53-72. DOI: 10.1016/0041-5553(80)90061-0.

Lugon, B. A. (2013). Algoritmos de reordenamento de matrizes esparsas aplicados a precondicionadores ILU(p). TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal do Espírito Santo, Vitória. 77 p.

Mehrotra, S. (1992). On The Implementation of a primal-dual interior point method. SIAM Journal on Optimization, Vol 2, nº 4, pp. 575-601. DOI: 10.1137/0802028.

Munari Jr, P. A. (2009). Comparação de *softwares* científicos utilizando perfis de desempenho: automatização dos cálculos pela planilha `perfis.xls`. Instituto de Ciências Matemáticas e de Computação Universidade de São Paulo - Campus de São Carlos. 9 p. Disponível em: <http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_344.PDF>. Acesso em: 10 nov. 2016.

Nicoletti, M. d. C. & Hruschka JR, E. R. (2006). Fundamentos da teoria dos grafos para computação - Série Apontamentos (Ed. Revisada). São Carlos: Editora EdUFSCar.

Oliveira, A. R. L. & Sorensen, D. C. (2005). A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. Linear Algebra and its Applications. Elsevier. vol. 394, p. 1-24. DOI: 10.1016/j.laa.2004.08.019.

Ross, C., Terlaky, T. & Vial, J.P. (2005). Interior Point Method for Linear Optimization (2º ed.). New York: Springer, 2005. 495 p. ISBN-13: 978-0387-26378-6.

Rothberg, E. & Hendrickson, B. (1998). Sparse matrix ordering methods for interior point linear programming. INFORMS Journal on Computing, vol. 10, nº. 1, pp. 107-113. DOI: 10.1287/ijoc.10.1.107.

Ruggiero, M. A. G. & Lopes, V.L.R. (1996). Cálculo numérico. Aspectos Teóricos e Computacionais (2º ed.). São Paulo: Pearson, 410 p. ISBN: 9788534602044.

Saad, Y. (2003). Iterative Methods for Sparse Linear Systems (2nd. Ed). SIAM - Society for Industrial and Applied Mathematics, 528p. ISBN: 978-0-89871-534-7.

Silva, D. C. (2015). Influência do Reordenamento de Matrizes no Desempenho de Métodos de Pontos Interiores para Programação Linear. Programa de Doutorado em Engenharia Elétrica, Universidade Estadual de Campinas, 129p.

Sloan, S. W. (1986). An Algorithm for Profile and Wavefront Reduction of Sparse Matrices. Vol. 23, pp. 239-251. DOI: 10.1002/nme.1620230208.

Tinney, W. F., Walker, J. W. (1967). Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization. Proceedings of the IEEE. Vol. 55, nº 11, pp. 1801-1809. DOI**:** 10.1109 / PROC.1967.6011.

Vanderbei, R. J. (2014). Linear Programming: Foundations and Extensions (4th Ed.). New York: Springer, 414 p. ISBN 978-1-4614-7630-6.

Velazco, M. I. & Oliveira, A. R. L. (2010). A note on hybrid preconditioners for large-scale normal equations arising from interior-point methods. The 22nd European Conference on Operational Research. Journal Optimization Methods and Software. Vol. 25, nº 2, pp. 331-332. DOI 10.1080/10556780902992829.

Yannakakis, M. (1981). Computing the Minimum Fill-in is NP-Complete. SIAM Journal on Algebraic Discrete Methods. Vol. 2, nº 1, pp. 77-79, DOI 10.1137/0602010.