Métodos de Busca em Coordenada

Luiz Gustavo de Moura dos Santos

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Ernesto G. Birgin

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CNPq

São Paulo, Janeiro de 2018

Métodos de Busca em Coordenada

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 22/11/2017. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo

Banca Examinadora:

- Prof. Dr. Ernesto G. Birgin (orientador) - IME-USP
- Prof. Dr. Walter F. Mascarenhas - IME-USP
- Prof. Dr. Luis Felipe Bueno - UNIFESP

Resumo

Problemas reais em áreas como aprendizado de máquina têm chamado atenção pela enorme quantidade de variáveis ($> 10^6$) e volume de dados. Em problemas dessa escala o custo para se obter e trabalhar com informações de segunda ordem são proibitivos. Tais problemas apresentam características que podem ser aproveitadas por *métodos de busca em coordenada*. Essa classe de métodos é caracterizada pela alteração de apenas uma ou poucas variáveis a cada iteração. A variante do método comumente descrita na literatura é a minimização cíclica de variáveis. Porém, resultados recentes sugerem que variantes aleatórias do método possuem melhores garantias de convergência. Nessa variante, a cada iteração, a variável a ser alterada é sorteada com uma probabilidade preestabelecida não necessariamente uniforme.

Neste trabalho estudamos algumas variações do método de busca em coordenada. São apresentados aspectos teóricos desses métodos, porém focamos nos aspectos práticos de implementação e na comparação experimental entre variações do método de busca em coordenada aplicados a diferentes problemas com aplicações reais.

Palavras-chave: Otimização convexa, busca em coordenada, busca em coordenada aleatória, *pagerank*, máquina de suporte vetorial.

Abstract

Real world problemas in areas such as machine learning are known for the huge number of decision variables ($> 10^6$) and data volume. For such problems working with second order derivatives is prohibitive. These problems have properties that benefits the application of *coordinate descent/minimization methods*. These kind of methods are defined by the change of a single, or small number of, decision variable at each iteration. In the literature, the commonly found description of this type of method is based on the cyclic change of variables. Recent papers have shown that randomized versions of this method have better convergence properties. This version is based on the change of a single variable chosen randomly at each iteration, based on a fixed, but not necessarily uniform, distribution. In this work we present some theoretical aspects of such methods, but we focus on practical aspects.

Palavras-chave: Convex optimization, coordinate minimization, random coordinate descent, *pagerank*, support vector machine.

Sumário

Lista de Tabelas

Lista de Figuras

Parte I

Fundamentação

Capítulo 1

Introdução

Neste trabalho estamos interessados em estudar problemas de otimização com grande número de variáveis. Problemas deste tipo aparecem comumente em áreas como processamento de imagens, processamento de sinais, resolução de equações diferenciais, aprendizado de máquina, entre outras. Especificamente, consideraremos o problema irrestrito de otimização dado por

$$\min_{x\in\mathbb{R}^n} f(x).$$

Os métodos de otimização que podem ser aplicados a este problema seguem, basicamente, dois paradigmas possíveis: busca linear e regiões de confiança [21]. Os métodos que estudaremos neste trabalho encaixam-se no primeiro caso. Os algoritmos de busca linear geram uma sequência de iterados x^k da forma

$$x^{k+1} = x^k + t_k d^k\,, \tag{1.1}$$

onde d^k é uma direção de busca e t_k é o tamanho do passo. No método de máxima descida utiliza-se $d^k = -\nabla f(x^k)$. Para obter propriedades teóricas de convergência o passo t_k pode ser calculado como o maior elemento do conjunto $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots\}$ que satisfaz a equação

$$f(x^k + t_k d^k) \leq f(x_k) + \sigma t_k \left\langle \nabla f(x^k), d^k \right\rangle, \tag{1.2}$$

onde $\sigma \in (0,1)$ é um parâmetro do método. A condição (1.2) é conhecida como *condição de Armijo* ou condição de decréscimo suficiente. Uma forma de calcular t_k é começar com $t = 1$ e, enquanto

$$f(x^k + td^k) > f(x_k) + t\gamma \left\langle \nabla f(x^k), d^k \right\rangle\,, \tag{1.3}$$

dividir t por 2. Quando a condição (1.3) é satisfeita definimos $t_k = t$. Provas de convergência deste método podem ser encontradas em qualquer livro clássico de otimização [15]. Em um algoritmo de busca linear como o acima descrito as operações mais custosas em cada iteração são a avaliação do gradiente $\nabla f(x^k)$ e a avaliação da função objetivo f nos pontos $x^k + td^k$. Quando o número de variáveis é muito grande o custo destas operações pode ser proibitivo.

Os métodos de busca em coordenadas [3, 19, 21] tentam aliviar estes custos. Nessa família de métodos em cada iteração k altera-se uma coordenada i do vetor x^k buscando minimizar a função objetivo, enquanto as demais coordenadas permanecem inalteradas. Na equação (1.1) a direção seria da forma $d^k = \pm e_i$, onde e_i é o i-ésimo vetor da base canônica de $\mathbb{R}^n$.

Os métodos de busca em coordenadas têm potencial para serem aplicados em problemas nos quais avaliar uma única ou umas poucas derivadas parciais da função objetivo seja uma tarefa barata ou em problemas em que a função objetivo seja separável de forma tal que, se já temos o valor de f num ponto x e modificamos umas poucas componentes de x, reavaliar a função objetivo no novo ponto seja uma tarefa barata.

Vejamos o exemplo dado por Nesterov [19]. Considere o problema de minimização irrestrita

$$\min_{x\in\mathbb{R}^n} f(x) \stackrel{def}{=} \frac{1}{2}\left\|Ax - b\right\|^2 + \sum_{i=1}^{n} h_i(x_i), \tag{1.4}$$

onde $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ e h_i é uma função contínua e diferenciável de uma variável. O cálculo de uma componente do gradiente f é dado por

$$[\nabla f(x)]_i = \langle A_i, r(x)\rangle + h_i'(x_i),$$

onde $r(x) = Ax - b$ e A_i é a i-ésima coluna de A. Se o vetor residual $r(x)$ já está calculado então obter a i-ésima componente do gradiente de f requer $O(p_i)$ operações, onde p_i é o número de elementos não nulos do vetor A_i. Por sua vez, o movimento $x' = x + \alpha e_i$, resulta na seguinte alteração do vetor residual:

$$r(x') = r(x) + \alpha A_i.$$

Sendo assim, o passo na i-ésima coordenada para o problema (1.4) requer $O(p_i)$ operações. Note que calcular tanto o valor de f quanto seu gradiente em algum ponto exige $O(\sum_{i=1}^{n} p_i)$ operações.

A escolha da coordenada a ser minimizada pode ser feita de diversas maneiras, sendo uma delas a escolha cíclica de coordenada. Vários livros de otimização abordam essa estratégia [1, 4, 15, 22] que consiste em alterar as coordenadas na ordem dada pela sequência $\{1, 2, \ldots, n, 1, \ldots\}$. Apesar dos primeiros trabalhos envolvendo escolha cíclica datarem da década de 50 e 60 [8,27] os primeiros resultados de convergência assintótica só foram publicados em 1992 [16] e resultados de convergência global, ainda mais recentes, em 2014 [24]. Esse interesse pode ser atribuído ao sucesso da aplicação do método a problemas de grande interesse. O recente trabalho de Stephen J. Wright [28] faz uma revisão de vários métodos de descida em coordenada, listando aplicações e resultados teóricos de convergência para diversos problemas.

No problema (1.4), tomando $h_i(x) = \lambda|x_i|, \lambda \in \mathbb{R}$, obtemos o problema conhecido como LASSO [25] (*Least Absolute Shrinkage and Selection Operator*). Podemos destacar o trabalho de Friedman et al. [9], onde métodos de busca em coordenada são aplicados ao LASSO e excelentes resultados são obtidos. O LASSO possui aplicações em diversas áreas como: reconstrução de imagens, tomografia por emissão de positrons, aprendizado de máquina, máquinas de suporte vetorial e *compressive sensing* (onde é chamado de *basis pursuit*).

O resto desse trabalho está organizado da seguinte maneira: O Capítulo 2 apresenta a forma geral do método e resultados de convergência. No Capítulo 3 estudamos a aplicação de algumas variantes do método de busca em coordenada aplicadas a um problema quadrático. No Capítulo 4 aplicamos o método de busca em coordenada ao problema de atribuir importância à páginas *web* e o comparamos ao algoritmo *pagerank*. No Capítulo 5 o método de busca em coordenada é aplicado ao treino de uma máquina de suporte vetorial linear para classificação de texto. Conclusões e considerações finais são apresentadas no Capítulo 6.

Capítulo 2

Método de Busca em Coordenada

2.1 Introdução

Considere o problema de minimização irrestrita

$$\min_x f(x)$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função contínua. A ideia do *método de busca em coordenada* é aproximar o iterando x^k à um minimizador de f alterando apenas uma das componentes do vetor x^k a cada iteração. A sequência de iterandos gerados por essa classe métodos é da forma

$$x^{k+1} = x^k + t_k e_{i_k}\,, \tag{2.1.1}$$

onde

$t_k \in \mathbb{R}$ é o tamanho do passo,

i_k é o índice da variável alterada durante a k-ésima iteração e

e_i é o i-ésimo vetor da base canônica de $\mathbb{R}^n$.

Estendendo essa ideia, o método de busca em *bloco* de coordenadas considera a alteração de um número arbitrário de variáveis por iteração. Por exemplo, a alteração simultânea das coordenadas 1 e 2 na k-ésima iteração teria a forma

$$x^{k+1} = x^k + t_{k,1}e_1 + t_{k,2}e_2\,. \tag{2.1.2}$$

A equação (2.1.2) pode ser simplificada com a introdução

$$\text{da matriz} \quad U = [e_1\ e_2] \quad \text{e do vetor} \quad t^k = (t_{k,1},\ t_{k,2})^T,$$

resultando na expressão

$$x^{k+1} = x^k + Ut^k. \tag{2.1.3}$$

Na Seção 2.2 é apresentada a generalização da ideia de *blocos de variáveis* para tamanhos arbitrário e são introduzidas notações que serão úteis na descrição dos métodos estudados nesse trabalho.

Diversas variações do método de busca em bloco de coordenadas são descritas na literatura. Neste trabalho examinaremos dois aspectos dessas variações. O primeiro deles diz respeito à maneira com são escolhidos os blocos que serão alterados a cada iteração. As estratégias de escolha relevantes neste trabalho são descritas na Seção 2.4. O segundo aspecto corresponde à alteração das variáveis do bloco escolhido. Na Seção 2.3 descrevemos as estratégias de atualização do bloco estudadas.

2.2 Notação

A notação aqui apresentada é semelhante à utilizada por Nesterov [19]. Consideraremos o problema

$$\min_{x\in\mathbb{R}^N} f(x) \tag{2.2.1}$$

e a partição $x = (x_1, x_2, \ldots, x_n)^T$ do vetor de variáveis x onde:

$$\begin{aligned} &n \quad \text{é o número de blocos,}\\ &x_i \in \mathbb{R}^{n_i} \quad \text{é o } i\text{-ésimo } \textit{bloco} \text{ de variáveis,}\\ &n_i \quad \text{é o } \textit{tamanho} \text{ do } i\text{-ésimo bloco e}\\ &n_1 + \cdots + n_n = N. \end{aligned}$$

De maneira análoga podemos particionar a matriz identidade da seguinte maneira,

$$I_N = \begin{pmatrix} U_1 & U_2 & \cdots & U_n \end{pmatrix}, \quad \text{onde} \quad U_i \in \mathbb{R}^{N\times n_i}, \quad i = 1, \ldots, n.$$

Note que

$$x = \sum_{i=1}^{n} U_i x_i \quad \text{e} \quad x_i = U_i^T x.$$

Podemos supor sem perda de generalidade que os blocos são compostos por variáveis consecutivas. Basta uma permutação apropriada de x para que qualquer partição satisfaça tal propriedade. Os iterandos do método de busca em *bloco* de coordenadas têm a forma

$$x^{k+1} = x^k + U_{i_k} u^k\,, \tag{2.2.2}$$

onde u^k é o vetor de passos. Quando f for diferenciável denotaremos por $\nabla_{x_i} f(x)$ a derivada parcial de f em relação ao i-ésimo bloco de variáveis, isto é,

$$\nabla_{x_i} f(x) = \frac{\partial f(x)}{\partial x_i} = U_i^T \nabla f(x) \tag{2.2.3}$$

e denotaremos

$$g^k = \nabla f(x^k) \quad \text{e} \quad g_i^k = \nabla_{x_i} f(x^k),\ i = 1, \ldots, n.$$

Se f for duas vezes diferenciável denotaremos por $\nabla^2_{x_i} f(x) \in \mathbb{R}^{n_i\times n_i}$ o bloco diagonal da matriz Hessiana de f correspondente ao i-ésimo bloco de variáveis, dado por

$$\nabla^2_{x_i} f(x) \;=\; \frac{\partial^2 f(x)}{\partial x_i \partial x_i} \;=\; U_i^T \nabla^2 f(x) U_i\,. \tag{2.2.4}$$

Também denotaremos

$$H^k = \nabla^2 f(x^k) \quad \text{e} \quad H_i^k = \nabla^2_{x_i} f(x^k),\ i = 1, \ldots, n.$$

2.2.1 Bloco Lipschitz

O gradiente ∇f é dito *Lipschitz contínuo* se satisfaz a inequação

$$\|\nabla f(x) - \nabla f(y)\| \le L\,\|x - y\|\,, \tag{2.2.5}$$

para todo x e y no domínio de f e alguma constante $L > 0$. De maneira semelhante, dizemos que ∇f possui *blocos Lipschitz* contínuos se

$$\|\nabla_{x_i} f(x) - \nabla_{x_i} f(y)\| \le L_i\,\|x - y\|\,, \quad i = 1, \ldots, n\,, \tag{2.2.6}$$

para todo x e y no domínio de f distintos somente no i-ésimo bloco, isto é, $x - y = U_i u$ para algum $u \in \mathbb{R}^{n_i}$.

2.3 Alteração do bloco

Consideraremos duas estratégias na escolha do vetor de passos u^k. A primeira delas, mais frequente na literatura [3, 15, 21], é a escolha ótima, nesse caso o vetor de passos é

$$u^k = \arg\min_u \ f(x^k + U_{i_k} u)\,. \tag{2.3.1}$$

Essa estratégia requer que o problema (2.3.1) possa ser resolvido de forma fácil. Dependendo das propriedades de f tal exigência é razoável, principalmente ao considerarmos blocos de tamanho unitário. Uma característica dessa estratégia é que o problema (2.3.1) não requer diferenciabilidade de f. Por isso a busca em coordenada é frequentemente associada à problemas de função objetivo não diferenciável ou cujo calculo do vetor gradiente seja de elevado custo computacional.

Se f é quadrática e convexa o vetor de passos u^k pode ser obtido resolvendo o sistema

$$H^k_{i_k} u^k = -g^k_{i_k}, \tag{2.3.2}$$

exigindo nesse caso o cálculo de $H^k_{i_k}$, $g^k_{i_k}$ e $O(n^3_{i_k})$ operações para a resolução do sistema (2.3.2). Se $n_{i_k} = 1$ então calcular o vetor u^k corresponde ao calculo de uma entrada da matriz Hessiana de f e uma componente do vetor gradiente. Ainda para o caso de f quadrática, se $n_{i_k} > 1$ o custo por iteração $O(n^3_{i_k})$ pode ser amenizado com a fatoração antecipada das matrizes $H^k_{i_k}$ reduzindo o custo da resolução do sistema (2.3.2) para $O(n^2_{i_k})$.

A outra estratégia que consideraremos, e que tem recebido mais atenção em trabalhos recentes [2, 19, 24], consiste em dar um passo de tamanho fixo na direção oposta ao gradiente do bloco. O passo é definido da seguinte maneira

$$u^k = -\frac{1}{L_{i_k}} g^k_{i_k}\,, \tag{2.3.3}$$

onde L_i é a *constante Lipschitz* associada ao i-ésimo bloco (descrita na Seção 2.2.1).

Por convenção, nos referiremos à estratégia (2.3.1) por *minimização* de coordenada e à (2.3.3) por *descida* de coordenada. Lembrando que não há uniformidade no uso desses termos na literatura.

2.4 Escolha do Bloco

Talvez a estratégia mais simples e mais estudada para a seleção do bloco a ser alterado seja a *cíclica*, onde os blocos são percorridos em sequência do primeiro ao último. Formalmente, os termos da sequência de índices de blocos são

$$i_0 = 1,$$
$$i_k = \begin{cases} 1 & \text{se } i_{k-1} = n \\ i_{k-1} + 1 & \text{caso contrário}\,. \end{cases}$$

Outra estratégia lembrada com frequência é a *Gauss-Southwell*, que consiste em escolher o bloco cuja norma do gradiente seja de maior valor

$$i_k = \arg\max_{i \in \{1,\ldots,n\}} \left\| g^k_i \right\|\,. \tag{2.4.1}$$

A estratégia Gauss-Southwell é normalmente citada pelas suas características teóricas. Porém, na prática, a necessidade do vetor gradiente completo a cada iteração impõe um custo muito alto para sua aplicação.

A terceira estratégia que consideraremos é a escolha aleatória do bloco de coordenadas. No método descrito por Nesterov [19], o bloco alterado na k-ésima iteração é escolhido com probabilidade

$$Prob(i_k = i) = \frac{L_i^\alpha}{\sum_{j=1}^{n} L_j^\alpha}, \quad i = 1, \ldots, n, \tag{2.4.2}$$

onde $\alpha \in [0,1]$ é um parâmetro que controla a relevância das constantes L_i para a escolha dos blocos. Se $\alpha = 0$ temos

$$Prob(i_k = i) = \frac{1}{n}, \quad i = 1, \ldots, n, \tag{2.4.3}$$

ou seja, os blocos são escolhidos com probabilidade igual e uniforme. Quando $\alpha = 1$ a probabilidade de escolha do bloco i é

$$Prob(i_k = i) = \frac{L_i}{\sum_{j=1}^{n} L_j}, \quad i = 1, \ldots, n, \tag{2.4.4}$$

ou seja, os blocos cujas constantes L_i possuem maior valor são escolhidos com mais frequência.

2.5 Convergência

Descrevemos nessa Seção resultados apresentados por Nesterov [19] para taxa convergência do método de busca em coordenada aleatória. Alguns detalhes omitidos em [19] são explicitados aqui enquanto que algumas generalizações são descartadas ou postergadas em benefício da simplicidade. Uma dessas generalizações diz respeito a norma: os resultados em [19] são dados em relação à normas arbitrárias, nas demonstrações a seguir $\|\cdot\|$ denota a norma euclidiana.

Para os resultados dessa Seção considere uma partição de $\mathbb{R}^N$ como descrita na Seção 2.2 e uma função f satisfazendo (2.2.6) para tal partição. Para a norma euclidiana temos

$$\sum_{i=1}^{n} \|x_i\|^2 = \|x\|^2 . \tag{2.5.1}$$

Seja $S = \sum_{i=1}^{n} L_i$, temos

$$\begin{aligned} \|\nabla f(x) - \nabla f(y)\| &= \left(\sum_{i=1}^{n} \|\nabla_i f(x) - \nabla_i f(y)\|^2 \right)^{1/2} \\ &\leq \sum_{i=1}^{n} \|\nabla_i f(x) - \nabla_i f(y)\| \\ &\leq \sum_{i=1}^{n} L_i \|x_i - y_i\| \\ &\leq \sum_{i=1}^{n} S \|x_i - y_i\| = S \|x - y\| . \end{aligned} \tag{2.5.2}$$

portanto ∇f é Lipschitz. O seguinte resultado para funções de gradiente Lipschitz será relevante para as demonstrações [20, Seção 2.1]:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{S}{2} \|x - y\|^2 . \tag{2.5.3}$$

O lema a seguir apresenta resultado semelhante para funções de gradiente bloco-Lipshitz.

Lema 1. *Para a função f temos que*

$$f(x + U_i d_i) \leq f(x) + \langle \nabla_i f(x),\, d_i \rangle + \frac{L_i}{2} \|d_i\|^2 , \tag{2.5.4}$$

onde $d_i \in \mathbb{R}^{n_i}$, para $i = 1, \ldots, n$.

Demonstração. Como f é contínua e diferenciável temos

$$\begin{aligned}
f(x+U_i d_i) &= f(x) + \int_0^1 \langle \nabla f(x+tU_i d_i), U_i d_i\rangle \, dt \\
&= f(x) + \int_0^1 \langle \nabla_i f(x+tU_i d_i), d_i\rangle \, dt \\
&= f(x) + \int_0^1 \langle \nabla_i f(x+tU_i d_i) - \nabla_i f(x) + \nabla_i f(x), d_i\rangle \, dt \\
&= f(x) + \langle \nabla_i f(x), d_i\rangle + \int_0^1 \langle \nabla_i f(x+tU_i d_i) - \nabla_i f(x), d_i\rangle \, dt \\
&\leq f(x) + \langle \nabla_i f(x), d_i\rangle + \int_0^1 \|\nabla_i f(x+tU_i d_i) - \nabla_i f(x)\| \, \|d_i\| \, dt \\
&\leq f(x) + \langle \nabla_i f(x), d_i\rangle + \int_0^1 tL_i \|d_i\|^2 \, dt \\
&= f(x) + \langle \nabla_i f(x), d_i\rangle + \frac{L_i}{2}\|d_i\|^2 .
\end{aligned}$$

□

O lema a seguir é extraído de uma das demonstrações em [19] e é apresentado separadamente para melhor entendimento.

Lema 2. *Se $\{x^k\}_{k\geq 0}$, $x^k \in \mathbb{R}$, é uma sequência monótona decrescente, limitada inferiormente por x^* e satisfazendo*

$$x^k - x^{k+1} \geq \frac{1}{\alpha}(x^k - x^*)^2, \tag{2.5.5}$$

para algum $\alpha > 0$, então

$$\lim_{k\to\infty} x^k = x^* \qquad com \qquad x^k - x^* \leq \frac{\alpha\beta}{\beta k + \alpha},$$

para uma constante $\beta \geq x^0 - x^*$.

Demonstração. Da inequação (2.5.5) temos

$$\frac{1}{\alpha} \;\leq\; \frac{x^k - x^{k+1}}{(x^k - x^*)^2} \;\leq\; \frac{x^k - x^{k+1}}{(x^k - x^*)(x^{k+1} - x^*)} \;=\; \frac{1}{x^{k+1} - x^*} - \frac{1}{x^k - x^*}.$$

Somando os primeiros $k+1$ termos de ambos os lados da inequação anterior obtemos

$$\sum_{j=0}^{k} \frac{1}{\alpha} \;=\; \frac{k+1}{\alpha} \quad\leq\quad \sum_{j=0}^{k} \left[\frac{1}{x^{j+1} - x^*} - \frac{1}{x^j - x^*}\right] \;=\; \frac{1}{x^{k+1} - x^*} - \frac{1}{x^0 - x^*}.$$

Assim

$$x^k - x^* \;\leq\; \left(\frac{k}{\alpha} + \frac{1}{x^0 - x^*}\right)^{-1} \;\leq\; \left(\frac{k}{\alpha} + \frac{1}{\beta}\right)^{-1} \;=\; \frac{\alpha\beta}{\beta k + \alpha}$$

□

Denominaremos por BD o método formado da seleção de bloco (2.4.4) e pela direção (2.3.3) (correspondente ao método RCDM(1, x^0) em [19]) aplicado ao problema $\min_x f(x)$. Seja X^* o conjunto solução (presumimos que X^* não é vazio), x^k o k-ésimo iterando na execução do método BD e

$$R = \max\{\, \|x - x^*\| \;:\; f(x) \leq f(x^0),\; x^* \in X^* \,\}.$$

Denotaremos

$$f^* = f(x^*), \quad f^k = f(x^k), \quad g^k = \nabla f(x^k) \quad \text{e} \quad g_i^k = U_i^T g^k \quad \text{para } i = 1, \ldots, n.$$

O valor esperado de f^{k+1} em relação à k-ésima iteração será denotado por $\mathbb{E}(f^k)$ e por ϕ^{k+1} o valor esperado em relação à todas as iterações anteriores.

Teorema 1. *Considere o valor esperado da sequência f^k gerada pela execução do método BD, temos que*

$$\phi^k - f^* \leq \frac{2SR^2}{k+4}. \tag{2.5.6}$$

Demonstração. Do Lema 1 temos

$$f^{k+1} \leq f^k - \frac{1}{L_{i_k}} \left\| g_{i_k}^k \right\|^2 + \frac{1}{2L_{i_k}} \left\| g_{i_k}^k \right\|^2 \;=\; f^k - \frac{1}{2L_{i_k}} \left\| g_{i_k}^k \right\|^2.$$

Tomando o valor esperado de ambos os lados da inequação em relação à k-ésima iteração obtemos

$$\begin{aligned} f^k - \mathbb{E}(f^{k+1}) \;\geq\; \mathbb{E}\left(\frac{1}{2L_{i_k}} \left\| g_{i_k}^k \right\|^2 \right) \;&=\; \sum_{j=1}^{n} Prob(i_k = j) \, \frac{1}{2L_j} \left\| g_j^k \right\|^2 \\ &= \sum_{j=1}^{n} \frac{L_j}{S} \frac{1}{2L_j} \left\| g_j^k \right\|^2 \;=\; \frac{1}{2S} \left\| g^k \right\|^2, \end{aligned} \tag{2.5.7}$$

note que f^k não depende de i_k. Da convexidade de f temos

$$0 \leq \; f^k - f^* \;\leq\; \left\langle g^k, x^k - x^* \right\rangle \;\leq\; \left\| g^k \right\| \left\| x^k - x^* \right\| \;\leq\; \left\| g^k \right\| R.$$

Assim

$$f^k - \mathbb{E}(f^{k+1}) \;\geq\; \frac{1}{2SR^2} \left(f^k - f^* \right)^2,$$

tomando o valor esperado em relação à k primeiras iterações temos

$$\phi^k - \phi^{k+1} \;\geq\; \frac{1}{2SR^2} \left(\phi^k - f^* \right)^2$$

(o lado direito da inequação é obtido via desigualdade de Jensen). Da equação 2.5.3 temos

$$f^0 - f^* \;\leq\; \frac{S}{2} \left\| x^0 - x^* \right\|^2 \;\leq\; \frac{1}{2} SR^2. \tag{2.5.8}$$

Assim a sequência ϕ^k satisfaz as condições do Lema 2 para $\beta = \frac{1}{2}SR^2$ e $\alpha = 4\beta$, resultando na inequação enunciada. □

Suponha agora que f também seja fortemente convexa, isto é, satisfaz a condição

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \left\| y - x \right\|^2, \; x, y \in \mathbb{R}^N, \tag{2.5.9}$$

para algum parâmetro $\mu > 0$ dependente de f. Nesse caso a seguinte taxa de convergência pode ser obtida.

Teorema 2. *Se f é fortemente convexa com parâmetro de convexidade μ então*

$$\phi^k - f^* \leq \left(1 - \frac{\mu}{S} \right)^k \left(f^0 - f^* \right). \tag{2.5.10}$$

Demonstração. Minimizando ambos os lados da inequação (2.5.9) em y temos $y = x^*$ para o lado direito e $y = -\mu^{-1}\nabla f(x) + x$ para o lado esquerdo, resultando em

$$f(x) - f(x^*) \leq \frac{1}{2\mu} \|\nabla f(x)\|^2 . \tag{2.5.11}$$

Aplicando essa desigualdade à equação (2.5.7) obtemos

$$\begin{aligned} f^k - \mathbb{E}\left(f^{k+1}\right) &\geq \frac{\mu}{S}\left(f^k - f^*\right), \\ \mathbb{E}\left(f^{k+1}\right) &\leq f^k - \frac{\mu}{S}\left(f^k - f^*\right), \\ \mathbb{E}\left(f^{k+1}\right) - f^* &\leq f^k - f^* - \frac{\mu}{S}\left(f^k - f^*\right), \\ \mathbb{E}\left(f^{k+1}\right) - f^* &\leq \left(1 - \frac{\mu}{S}\right)\left(f^k - f^*\right). \end{aligned}$$

E por indução

$$\mathbb{E}\left(f^{k+1}\right) - f^* \leq \left(1 - \frac{\mu}{S}\right)^{k+1}\left(f^0 - f^*\right).$$

Tomando o valor esperado em relação às k primeiras iterações obtemos o resultado enunciado. □

Os resultados anteriores são relativamente intuitivos quando o método BD é considerado. Podemos generalizar os esses resultados para a escolha de bloco com probabilidade (2.4.2), para isso considere as normas

$$\|x\|_{[\alpha]} = \left(\sum_{i=1}^n L_i^\alpha \|x_i\|^2\right)^{1/2} \qquad \text{e} \qquad \|x\|_{[\alpha]}^* = \left(\sum_{i=1}^n \frac{1}{L_i^\alpha}\|x_i\|^2\right)^{1/2}. \tag{2.5.12}$$

Note que a desigualdade de Cauchy-Schwarz é satisfeita:

$$\begin{aligned} \|x\|_{[\alpha]} \cdot \|y\|_{[\alpha]}^* &= \left(\sum_{i=1}^n L_i^\alpha \|x_i\|^2\right)^{1/2}\left(\sum_{i=1}^n \frac{1}{L_i^\alpha}\|y_i\|^2\right)^{1/2} \\ &= \left(\sum_{i=1}^n \|x_i\|^2 \, \|y_i\|^2 + \sum_{i=1}^n \sum_{\substack{j=1, \\ j\neq i}}^n \frac{L_i^\alpha}{L_j^\alpha}\|x_i\|^2 \, \|y_j\|^2\right)^{1/2} \\ &\geq \left(\sum_{i=1}^n \|x_i\|^2 \|y_i\|^2\right)^{1/2} \\ &\geq \left(\sum_{i=1}^n \langle x_i, y_i\rangle^2\right)^{1/2} \geq \sum_{i=1}^n |\langle x_i, y_i\rangle| \geq |\langle x, y\rangle|, \quad x, y \in \mathbb{R}^N. \end{aligned}$$

O seguinte teorema lista os resultados dados em [19] que generalizam os resultados dos teoremas 1 e 2.

Teorema 3. *Seja*

$$R_\alpha = \max\{\, \|x - x^*\|_{[\alpha]} \,:\, f(x) \leq f(x^0),\ x^* \in X^* \,\} \qquad e \qquad S_\alpha = \sum_{i=1}^n L_i^\alpha.$$

Temos

$$\phi^k - f^* \leq \frac{2}{k+4} S_\alpha R_{1-\alpha}^2. \tag{2.5.13}$$

Se f é fortemente convexa com respeito à norma $\|\cdot\|_{[1-\alpha]}$ com parâmetro $\mu_{1-\alpha}$, isto é, satisfaz

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_{1-\alpha}}{2} \|x - y\|^2_{[1-\alpha]}, \ x, y \in \mathbb{R}^N,$$

então

$$\phi^k - f^* \leq \left(1 - \frac{\mu_{1-\alpha}}{S_\alpha}\right)^k \left(f^0 - f^*\right). \tag{2.5.14}$$

Note que para $\alpha = 1$ os resultados correspondem aos dados nos teoremas 1 e 2. Nestes mesmos teoremas a simples aplicação das definições (2.5.12) fornecem os resultados do Teorema 3. A única observação é em relação à utilização da equação (2.5.2) para definir β no Teorema 2. Para obtermos os resultados do Teorema 3 a seguinte inequação deve ser considerada, em vez da equação (2.5.2):

$$\|\nabla f(x) - \nabla f(y)\|^*_{[1-\alpha]} \leq S_\alpha \|x - y\|_{[1-\alpha]}$$

e portanto

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2} S_\alpha \|x - y\|^2_{[1-\alpha]}.$$

A demonstração para o resultado anterior, que é mais elaborada do que a apresentada para (2.5.2), pode ser encontrada em [19].

Parte II

Aplicações

Capítulo 3

Mínimos Quadrados Linear

Neste capítulo analisaremos os métodos descritos no Capítulo 2 aplicados ao problema de mínimos quadrados linear. O problema é definido da seguinte maneira

$$\min_{x\in\mathbb{R}^N} \; f(x) = \frac{1}{2}\left\|Ax-b\right\|^2 , \tag{3.1}$$

consideraremos o caso em que a matriz A é quadrada de ordem N e não-singular, sob tais condições o problema (3.1) tem solução única. Na Seção 3.3 consideramos o caso em que A é sobredeterminada. Suporemos também que A é uma matriz esparsa com z elementos não nulos e uma média de p elementos não nulos por coluna e $pn = z$.

Como descrito na Seção 2.2 consideraremos a partição do vetor de variáveis

$$x = (x_1,\ x_2,\ \ldots,\ x_n)^T ,$$

onde

$$x_i \in \mathbb{R}^{n_i} \quad \text{e} \quad \sum_{i=1}^{n} n_i = N,$$

e também a partição da matriz identidade de ordem N,

$$I_N = (U_1\ U_2\ \cdots\ U_n) , \quad U_i \in \mathbb{R}^{N\times n_i} \quad \text{para}\ \ i = 1,\ldots,n.$$

Será útil também considerar a partição análoga da matriz A,

$$A = (A_1\ A_2\ \cdots\ A_n) \quad \text{onde} \quad A_i = AU_i \quad \text{para } i = 1,\ldots,n.$$

O vetor gradiente e a matriz Hessiana de f são dados respectivamente por

$$\nabla f(x) = A^T(Ax-b) \quad \text{e} \quad \nabla^2 f(x) = A^TA.$$

Denotaremos $Q = A^TA$. O gradiente e matriz Hessiana de f em relação ao i-ésimo bloco de variáveis são dados, respectivamente, por

$$\nabla_{x_i} f(x) = A_i^T(Ax-b) \quad \text{e} \quad \nabla^2_{x_i} f(x) = A_i^TA_i.$$

De maneira semelhante, denotaremos $\ Q_i = U_i^TQU_i = A_i^TA_i,\ \ Q_i \in \mathbb{R}^{n_i\times n_i}$.

Constantes Lipschitz

Nesta seção determinaremos o valor de constantes Lipschitz (Seção 2.2.1) associadas aos blocos de variáveis do problema (3.1). Por definição, a constante Lipschitz associada ao i-ésimo bloco é

$$\left\|\nabla_{x_i} f(x+U_ih) - \nabla_{x_i} f(x)\right\| \le L_i \left\|h\right\| . \tag{3.2}$$

Temos que

$$\begin{aligned}\nabla_{x_i} f(x + U_i h) &= A_i^T \left[A(x + U_i h) - b\right] \\ &= A_i^T (Ax - b) + A_i^T A_i h \\ &= \nabla_{x_i} f(x) + Q_i h.\end{aligned}$$

Portanto

$$\frac{\|Q_i h\|}{\|h\|} \leq L_i. \tag{3.3}$$

A definição da *norma matricial induzida* é

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}. \tag{3.4}$$

Assim, por definição, $L_i = \|Q_i\|$ satisfaz a inequação (3.2). Se $\|\cdot\|$ é a norma Euclidiana e como Q_i é positiva definida então $\|Q_i\| = \lambda_{max}$, onde λ_{max} é o maior autovalor da matriz Q_i. Nesses caso, $\|Q_i\|$ é, também, o menor valor possível para o qual a inequação (3.2) é satisfeita, já que $\|Q_i h\| = \|Q_i\| \, \|h\|$ se h é um autovetor associado à λ_{max}. Se $n_i = 1$ então A_i corresponde à i-ésima coluna da matriz A e portanto $L_i = \|A_i\|^2$.

3.1 Implementações

Além da sequência de iterandos x^k e de derivadas parciais $g_i^k,\ i = 1, \ldots, n$, (descrita na Seção 2.2) também será útil considerar os vetores residuais

$$r^k = Ax^k - b.$$

Para uma implementação eficiente da busca em coordenada para o problema (3.1) podemos aproveitar a estrutura esparsa de A para atualizar o vetor residual com poucas operações. Temos que

$$\begin{aligned}r^{k+1} &= Ax^{k+1} - b \\ &= A(x^k + U_{i_k} u^k) - b \\ &= Ax^k - b + A_{i_k} u^k \\ &= r^k + A_{i_k} u^k.\end{aligned}$$

Assim, a atualização do vetor residual na k-ésima iteração requer, em média, $O(n_{i_k} p)$ operações.

Consideraremos quatro implementações do método de busca em coordenada aplicadas ao problema (3.1): a minimização em coordenadas cíclicas, descida em coordenadas cíclica e a descida em coordenada aleatória uniforme e enviesada. Nas seções seguintes esses algoritmos são descritos e seus custos analisados. É recomendada a leitura da Seção 3.1.1 antes das demais, pois essa introduz algumas notações que serão utilizadas na descrição dos algoritmos subsequentes.

3.1.1 Minimização cíclica

Denominaremos por *minimização cíclica de (bloco de) coordenada* (*Cyclic Coordinate Minimization*, CCM) a combinação da estratégia cíclica de seleção de bloco com a escolha ótima do passo. Os passos são descritos em detalhes no Algoritmo CCMQ.

Algoritmo CCMQ(*Minimização cíclica de (bloco de) coordenada*). Os dados de entrada são: a matriz A e o vetor b; o ponto inicial x^0; os tamanhos $n_i,\ i = 1, \ldots, n$, de cada bloco.

1. [Inicialização]. São inicializados os valores

$$k \leftarrow 0, \quad i_0 = 1 \quad \text{e} \quad r^0 \leftarrow Ax^0 - b.$$

São calculadas as matrizes $Q_i \leftarrow A_i^T A_i$ para $i = 1, \ldots, n$, e a fatoração de Cholesky de cada matriz Q_i que não for diagonal.

2. [Escolha coordenada]. Se $k = 0$ vá para o passo 3. Senão, faça

$$i_k \leftarrow i_{k-1} + 1 \ \text{ se } \ i_{k-1} < n \quad \text{ou} \quad i_k = 1, \ \text{caso contrário.}$$

3. [Calculo do passo]. Calcule os vetores

$$g_{i_k}^k \leftarrow A_{i_k}^T r^k \quad \text{e} \quad u^k = -Q_{i_k}^{-1} g_{i_k}^k.$$

4. [Próximo iterando]. Os próximos iterandos são calculados de acordo com as expressões

$$x^{k+1} \leftarrow x^k + U_{i_k} u^k \qquad \text{e} \qquad r^{k+1} \leftarrow r^k + A_{i_k} u^k.$$

5. [Teste de parada]. Se x^{k+1} atende ao critério de parada então pare, caso contrário incremente k e volte para o Passo 2. ∎

Custo de inicialização

Considere o caso em que $n_i = m$, $i = 1, \ldots, n$, com $nm = N$, e nenhuma das matrizes Q_i é diagonal. Nesse caso, o Passo 1 do Algoritmo CCMQ requer $O(Nmp + Nm^2)$ operações.

O termo Nmp corresponde ao cálculo das matrizes Q_i. Cada entrada da matriz Q corresponde ao produto interno entre duas colunas da matriz A. Sendo assim, são necessárias $O(p)$ operações por entrada da matriz Q. No total temos n matrizes com m^2 entradas cada. Mesmo aproveitando a simetria das matrizes Q_i ainda é necessário o calculo de $O(m^2)$ elementos. Assim, são necessárias $O(nm^2p) = O(Nmp)$ operações.

O outro termo, Nm^2, corresponde à fatoração das matrizes Q_i. São realizadas n fatorações, já que estamos considerando o caso em que todas as matrizes Q_i são não-diagonais. Cada fatoração requer $O(m^3)$ operações, totalizando $O(nm^3) = O(Nm^2)$ operações.

Considerando o outro extremo, se toda matriz Q_i é diagonal então nenhuma fatoração é necessária. Intuitivamente, a probabilidade desse ser o caso é grande se $m < p \ll N$. Nesse caso o termo Nm^2 desaparece e o custo do Passo 1 passa a ser $O(Nmp)$.

O cálculo do vetor r^0 requer $O(Np)$ operações se $x^0 \neq 0$ e $O(N)$ se $x^0 = 0$. Em ambos os casos o custo do calculo de r^0 é dominado pelas outras operações descritas.

Finalmente, para o caso em que os blocos são de tamanho unitário o custo de inicialização é de $O(Np)$ operações.

Custo por iteração

Novamente suporemos que todos os blocos possuem tamanho m e nenhuma matriz Q_i é diagonal. Nesse caso cada iteração do Algoritmo CCMQ requer $O(mp + m^2)$ operações.

Uma iteração do Algoritmo CCMQ é composta pelos Passos 2, 3, 4, e 5 executados exatamente uma vez. O Passo 2 é trivial, o mesmo será assumido sobre o Passo 5. O termo mp corresponde ao calculo do vetor $g_{i_k}^k$. Enquanto que o termo m^2 é devido à resolução do sistema linear $Q_{i_k} u^k = -g_{i_k}^k$ utilizando a fatoração da matriz Q_{i_k} calculada no Passo 1. O Passo 4 requer $O(mp)$ operações devido o calculo do vetor r^{k+1}, conforme descrito na Seção 3.1.

Se todas as matrizes Q_i são diagonais então o calculo do vetor u^k pode ser feito com $O(m)$ operações. Nesse caso o custo por iteração é $O(mp)$ operações.

Se $m = 1$ o custo por iteração do Algoritmo CCMQ é de $O(p)$ operações.

3.1.2 Descida cíclica

Chamaremos de *Descida Cíclica em (bloco de) coordenada* (*Cyclic Coordinate Descent*, CCDQ) a combinação da escolha cíclica de blocos com o passo na direção oposta ao gradiente do bloco. Os detalhes da implementação são descritos no Algoritmo CCDQ.

Algoritmo CCDQ(*Busca em coordenada cíclica na direção do gradiente*). Os dados de entrada são: a matriz A e vetor b; o ponto inicial x^0; os tamanhos de blocos $n_i,\ i = 1, \ldots, n$.

1. [Inicialização]. São inicializados os valores

$$k \leftarrow 1, \quad i_0 \leftarrow 0 \quad \text{e} \quad r^0 \leftarrow Ax^0 - b.$$

São calculadas as matrizes $Q_i \leftarrow A_i^T A_i$ para $i = 1, \ldots, n$ e as constantes $L_i \leftarrow \|Q_i\|$, para $i = 1, \ldots, n$, pelo método das potências.

2. [Escolha coordenada]. $i_k \leftarrow i_{k-1} + 1$ se $i_k < n$ ou $i_k = 1$ caso contrário.

3. [Calculo do passo]. Calcule os vetores

$$g_{i_k}^k \leftarrow A_{i_k}^T r^k \quad \text{e} \quad u^k \leftarrow -\frac{1}{L_{i_k}} g_{i_k}^k.$$

4. [Próximo iterando]. Os próximos iterandos são calculados de acordo com as expressões

$$x^{k+1} \leftarrow x^k + U_{i_k} u^k \qquad \text{e} \qquad r^{k+1} \leftarrow r^k + A_{i_k} u^k.$$

5. [Teste de parada]. Se x^{k+1} atende ao critério de parada então pare, caso contrário incremente k e volte para o Passo 2. ∎

Custo de inicialização

Novamente consideramos o caso em que $n_i = m$, $i = 1, \ldots, n$, com $nm = N$ e nenhuma matriz das matrizes Q_i é diagonal. Nesse caso, o Passo 1 do Algoritmo CCMD requer $O(Nmp + nP(m, \epsilon))$ operações.

Assim como na inicialização do Algoritmo CCMQ, o termo Nmp corresponde ao calculo das matrizes Q_i. Porém, o Algoritmo CCMD não requer a fatoração dessas matrizes. Em vez disso, é necessário calcular o maior autovalor de cada matriz Q_i, correspondentes aos valores das constantes L_i. Para o calculo dos autovalores utilizamos o método das potências. O valor $P(m, \epsilon)$ corresponde à quantidade de operações necessárias para obtermos uma aproximação para maior autovalor de uma matriz de ordem m com precisão ϵ.

Alternativamente, se nenhuma matriz Q_i é diagonal o custo de inicialização é $O(Nmp)$. Pois os autovalores de uma matriz diagonal são os elementos da diagonal. Assim, se todas as matrizes Q_i forem diagonais a constante L_i pode ser obtida por simplesmente selecionando o maior elemento da diagonal de Q_i. Nesse caso o termo $nP(m, \epsilon)$ é substituído por nm, que é dominado pelo termo Nmp.

Se $m = 1$ então o custo de inicialização é de $O(Np)$ operações.

Custo por iteração

Assim como no Algoritmo CCMQ, o custo de uma iteração do Algoritmo CCDQ é dominado pelos Passos 3 e 4. Em ambos os Passos a quantidade de operações exigida é $O(mp)$, para o cálculo de $g_{i_k}^k$ no Passo 3 e r^{k+1} no Passo 4.

Então, o custo por iteração do Algoritmo CCDQ é $O(mp)$.

3.1.3 Descida aleatória

Como mencionado na Seção 2.4 a probabilidade de escolha de bloco apresentada por Nesterov [19] é

$$Prob(i_k = i) = \frac{L_i^\alpha}{\sum_{j=1}^n L_j^\alpha}, \quad i = 1, \ldots, n. \tag{3.1.1}$$

Consideraremos os valores $\alpha = 0$ e $\alpha = 1$. Chamaremos de *Descida aleatória uniforme em (bloco de) coordenada* (*Uniform Coordinate Descent*, UCD) o método obtido com $\alpha = 0$ e *Descida aleatória enviesada em (bloco de) coordenada* (*Biased Coordinate Descent*, BCD) quando $\alpha = 1$. O Algoritmo UCDQ/BCDQ apresenta ambos os métodos, os passos 1b e 2b são exclusivos para o método BCDQ, enquanto que o passo 2u é executado somente pelo UCDQ.

Algoritmo UCDQ/BCDQ(*Descida aleatória uniforme/enviesada em coordenada*). Os dados de entrada são: a matriz A e vetor b; o ponto inicial x^0; os tamanhos de blocos $n_i,\ i = 1, \ldots, n$.

1. [Inicialização]. São inicializados os valores

$$k \leftarrow 1, \quad i_0 \leftarrow 0 \quad \text{e} \quad r^0 \leftarrow Ax^0 - b.$$

São calculadas as matrizes $Q_i \leftarrow A_i^T A_i$ para $i = 1, \ldots, n$, e as constantes $L_i \leftarrow \|Q_i\|$, para $i = 1, \ldots, n$.

1b. [Inicialização BCDQ] É inicializada a tabela de distribuições substitutas.

2u. [Escolha uniforme]. Sorteie a coordenada i_k com probabilidade

$$Prob(i_k = i) = \frac{1}{n}, \quad i = 1, \ldots, n.$$

2b. [Escolha enviesada]. Sorteie a coordenada i_k com probabilidade

$$Prob(i_k = i) = \frac{L_i}{\sum_{j=1}^n L_j}, \quad i = 1, \ldots, n,$$

utilizando o algoritmo de distribuições substitutas.

3. [Calculo do passo]. Calcule os vetores

$$g_{i_k}^k \leftarrow A_{i_k}^T r^k \quad \text{e} \quad u^k \leftarrow -\frac{1}{L_{i_k}} g_{i_k}^k.$$

4. [Próximo iterando]. Os iterandos são atualizados de acordo com as expressões

$$x^{k+1} \leftarrow x^k + U_{i_k} u^k \qquad \text{e} \qquad r^{k+1} \leftarrow r^k + A_{i_k} u^k.$$

5. [Teste de parada]. Se x^k atende ao critério de parada então pare, caso contrário incremente k e volte para o Passo 2. ∎

Custo de inicialização

A inicialização do Algoritmo UCDQ é essencialmente a mesma do Algoritmo CCDQ, que como vimos requer $O(Nmp + nP(m, e))$ operações. O Algoritmo BCDQ o calculo adicional da tabela de distribuições substitutas (discutida logo mais), que requer $O(n)$ operações.

Custo por iteração

A diferença na iteração dos Algoritmos UCDQ e BCDQ para o CCDQ está unicamente na escolha do bloco a ser alterado. A escolha uniforme pode ser feita com $O(1)$ operações, porém a escolha enviesada (com $\alpha = 1$) requer um pouco mais de atenção. O algoritmo de sorteio sugerido por Nesterov [19] requer $O(\log n)$ operações por iteração, o que é um custo aceitável para a maioria das aplicações. Porém, com a quantidade de variáveis que gostaríamos de lidar o custo logarítmico pode dominar a iteração. O algoritmo utilizado para o sorteio das coordenadas é o método das distribuições substitutas (*Alias Method*) descrito no Apêndice A. A inicialização do AM requer $O(n)$ operações, após essa inicialização podemos sortear um bloco com a probabilidade desejada com $O(1)$ operações. Assim, o custo de inicialização é compensado após n iterações do Algoritmo BCDQ.

Então, apesar do sorteio não trivial do bloco de coordenada, o custo de iteração tanto do UCDQ quando BCDQ é de $O(pm)$ operações.

3.1.4 Máxima descida

Nas próximas seções serão apresentados resultados experimentais para as implementações dos métodos de busca em coordenada. Para comparação também foram implementados métodos gradientes para o problema (3.2.1). Os iterandos desses métodos têm a forma

$$x^{k+1} = x^k - \alpha_k g^k. \tag{3.1.2}$$

Temos que $g^k = A^T\left(Ax^k - b\right)$, assim sendo, os métodos descritos a seguir requerem no mínimo duas multiplicações matriz-vetor por iteração.

Minimização exata

Nesse caso α_k é escolhido de forma ótima em relação à direção g^k, ou seja

$$\alpha_k = \arg\min_t \ \phi_k(t), \quad \text{onde } \phi_k(t) = f(x^k - tg^k). \tag{3.1.3}$$

Como f é diferenciável, quadrática e convexa o solução pode ser obtida resolvendo a equação $\phi_k'(t) = 0$. A função ϕ_k e sua primeira derivada são

$$\phi_k(t) = \frac{1}{2}f(x) - t\left\|g^k\right\|^2 + \frac{t^2}{2}\left\|Ag^k\right\|^2 \quad \text{e} \quad \phi_k'(t) = -\left\|g^k\right\|^2 + t\left\|Ag^k\right\|^2,$$

logo

$$\alpha_k = \frac{\left\|g^k\right\|^2}{\left\|Ag^k\right\|^2}.$$

Assim o custo computacional para a minimização exata na direção g^k requer uma multiplicação matriz-vetor adicional, totalizando 3 multiplicações.

Armijo

No caso do problema (3.2.1) a minimização exata na direção g^k é facilmente encontrada. Porém para maioria dos problemas é mais vantajoso escolher um valor de α_k que satisfaça a condição de Armijo, dada por

$$f(x^k - tg^k) \leq f^k - t\sigma\left\|g^k\right\|^2, \tag{3.1.4}$$

onde $\sigma \in (0, 1)$ é o parâmetro que controla o decréscimo da função objetivo a cada iteração. Uma maneira de encontrar o valor t apropriado consiste em percorrer a sequência $\{\,1,\ 1/2,\ 1/4,\ \ldots\}$ e selecionar o primeiro termo que satisfaça a condição (3.1.4). Essa estratégia é conhecida como

backtracking. Cada passo de *backtraking* requer uma avaliação adicional da função objetivo, correspondendo à uma multiplicação matriz-vetor adicional.

No melhor caso, em que $t = 1$ satisfaz a condição (3.1.4), cada iteração requer duas multiplicações matriz-vetor, uma a menos em relação à minimização exata na direção g^k.

Espectral

Dependendo das propriedades da matriz A a quantidade de avaliações da função objetivo para satisfazer a condição de Armijo pode ser muito grande. No método gradiente espectral a direção $-g^k$ em (3.1.2) é substituída por $-\lambda_k g^k$, onde λ_0 é um valor arbitrário positivo e para $k > 0$

$$\lambda_k = \frac{\left\|s^k\right\|^2}{\langle s^k, y^k\rangle} \quad \text{onde} \quad s^k = x^k - x^{k-1} \quad \text{e} \quad y^k = g^k - g^{k-1}.$$

O passo α_k é então escolhido seguindo a mesma estratégia de *backtraking* do método anterior, porém visando satisfazer a condição

$$f(x^k - tg^k) \leq f^k_{max} - t\sigma \left\|g^k\right\|^2.$$

onde $f^k_{max} = \max\{ f^j : \min\{0, k - M + 1\} \leq j \leq k\}$, isto é, o decréscimo requerido para a função objetivo é em relação à M iterações anteriores. No caso do problema (3.2.1) temos

$$\lambda_k = \frac{\left\|g^{k-1}\right\|^2}{\left\|Ag^{k-1}\right\|^2},$$

ou seja, λ_k corresponde ao passo ótimo da equação (3.1.3) em relação a iteração $k-1$. Uma análise geral do método gradiente espectral pode ser encontrada em [5], uma discussão informal sobre o comportamento esperado pelo gradiente espectral aplicado ao problema (3.2.1) é dada a seguir.

Note que para o problema (3.2.1) temos

$$f(x^k - t\lambda_k g^k) - f^k = -t\lambda_k \left\|g^k\right\|^2 + t^2\lambda_k^2 \left\|Ag^k\right\|^2,$$

substituindo em (3.1.4) obtemos

$$t \;\leq\; (1-\sigma)\frac{1}{\lambda_k}\frac{\left\|g^k\right\|^2}{\left\|Ag^k\right\|^2} \;=\; (1-\sigma)\frac{\lambda_{k+1}}{\lambda_k}.$$

Note também que λ_k corresponde ao inverso do quociente de Rayleigh para matriz A^TA em relação ao vetor g^k, portanto $\lambda_{min} \leq 1/\lambda_k \leq \lambda_{max}$, onde λ_{min} e λ_{max} são, respectivamente, o menor e maior autovalor da matriz A^TA. Podemos então esperar que se $\lambda_{min} \approx \lambda_{max}$ e $\sigma \approx 0$ teremos $(1-\sigma)\frac{\lambda_{k+1}}{\lambda_k} \approx 1$, o que significa que são grandes as chances de que a condição (3.1.4) seja satisfeita sem necessidade de *backtracking* e ainda maiores de que um passo de *backtraking* seja suficiente.

3.2 Experimentos computacionais

3.2.1 Instâncias Aleatórias - Matrizes Quadradas

Foram geradas instâncias aleatórias para o problema

$$\min_x f(x) = \frac{1}{2}\left\|Ax - b\right\|^2, \tag{3.2.1}$$

onde $A \in \mathbb{R}^{N\times N}$. As instâncias são identificadas pela dimensão do problema (N) e número médio de elementos não-nulos por coluna (p). Foram geradas 12 instâncias com os parâmetros

$$(N,\, p) \in \{131072,\, 262144,\, 524288,\, 1048576\} \times \{16,\, 32,\, 64\}.$$

Para cada instância foram executados sete métodos iterativos. A Tabela 3.1 contém informações sobre as instâncias de acordo com o tamanho de bloco. A coluna "diag." contém a percentagem de matrizes Q_i que são diagonais. As colunas $\min L_i$ e $\max L_i$ contém o valor das constantes L_i de menor e maior valor, respectivamente. A relevância dessas informações é discutida mais a frente.

p	b	blocos	diag.	$\min L_i$	$\max L_i$
16	1	131072	100.0%	0.5	13.0
	2	65536	99.8%	2.1	13.0
	16	8192	79.4%	5.5	13.0
	64	2048	2.4%	7.3	13.0
32	1	131072	100.0%	2.9	21.2
	2	65536	99.2%	5.8	21.2
	16	8192	40.3%	11.1	21.2
	64	2048	0.0%	13.4	21.2
64	1	131072	100.0%	10.4	36.5
	2	65536	96.9%	14.0	36.5
	16	8192	2.4%	21.7	36.5
	64	2048	0.0%	24.8	36.5

(a) $N = 131072$

p	b	blocos	diag.	$\min L_i$	$\max L_i$
16	1	262144	100.0%	0.4	14.2
	2	131072	99.9%	1.5	14.2
	16	16384	89.5%	5.4	14.2
	64	4096	16.0%	7.4	14.2
32	1	262144	100.0%	2.6	22.5
	2	131072	99.6%	5.1	22.5
	16	16384	63.3%	10.7	22.5
	64	4096	0.0%	13.2	22.5
64	1	262144	100.0%	9.9	37.4
	2	131072	98.4%	13.3	37.4
	16	16384	15.5%	21.0	37.4
	64	4096	0.0%	25.1	37.4

(b) $N = 262144$

p	b	blocos	diag.	$\min L_i$	$\max L_i$
16	1	524288	100.0%	0.5	13.5
	2	262144	100.0%	1.7	13.5
	16	32768	94.4%	5.3	13.5
	64	8192	37.6%	7.1	13.5
32	1	524288	100.0%	2.5	22.9
	2	262144	99.8%	4.9	22.9
	16	32768	79.3%	10.4	22.9
	64	8192	2.2%	13.2	22.9
64	1	524288	100.0%	8.1	37.3
	2	262144	99.2%	12.8	37.3
	16	32768	39.3%	21.2	37.3
	64	8192	0.0%	24.5	37.3

(c) $N = 524288$

p	b	blocos	diag.	$\min L_i$	$\max L_i$
16	1	1048576	100.0%	0.4	15.7
	2	524288	100.0%	1.6	15.7
	16	65536	97.1%	5.3	15.7
	64	16384	62.2%	7.0	15.7
32	1	1048576	100.0%	2.4	22.7
	2	524288	99.9%	5.1	22.7
	16	65536	89.3%	10.3	22.7
	64	16384	14.6%	12.7	22.7
64	1	1048576	100.0%	7.8	36.9
	2	524288	99.6%	12.6	36.9
	16	65536	62.9%	21.4	36.9
	64	16384	0.0%	24.4	37.0

(d) $N = 1048576$

Tabela 3.1: Informações sobre os blocos obtidos em cada uma das instâncias de acordo com o tamanho de bloco escolhido

Método direto

Além dos métodos iterativos foi executada a rotina ma57 da biblioteca HSL [10] sobre um conjunto separado de instâncias. A rotina ma57 implementa um método direto para encontrar soluções aproximadas para o sistema o sistema linear $Ax = b$, onde A é simétrica. Foram geradas matrizes simétricas com quantidade de elementos não-nulos próximos à instâncias descritas na Seção 3.2.1. As execuções foram realizadas em uma máquina com 16GB de memória e processador Intel® Core™ i7-7700K 4.20GHz.

A Tabela 3.2 lista os resultados obtidos. A coluna *inst.* corresponde à memória (em MB) ocupada pela matriz A em formato COO. A coluna *mem.* contém a quantidade de memória requerida pela rotina ma57 para execução do método (contabilizando apenas os vetores mais relevantes). O tempo é dado em minutos nos valores em que a unidade não é especificada. A coluna f contém o valor da função objetivo (3.2.1) no ponto retornado pela rotina ma57.

n	p	inst.	mem.	tempo	f
1024	15	0.3	6.4	0.00	4.12e−28
	33	0.5	12.9	0.00	2.95e−27
	59	0.9	17.7	0.00	1.22e−26
2048	15	0.5	24.3	0.00	1.28e−27
	34	1.1	51.0	0.01	1.10e−26
	60	1.9	68.9	0.01	3.29e−26
4096	15	1.0	94.8	0.01	5.08e−27
	34	2.2	199.0	0.03	3.93e−26
	61	3.9	268.4	0.09	1.32e−25
8192	15	2.0	378.6	0.08	1.94e−26
	34	4.4	787.0	0.20	1.35e−25
	61	7.8	1053.8	0.69	4.78e−25
16384	15	4.1	1493.4	0.56	7.08e−26
	34	8.8	3116.6	1.45	5.39e−25
	61	15.5	4173.0	2.04	1.85e−24
32768	15	8.1	5965.7	4.31	2.75e−25
	34	17.6	12455.4	11.44	2.21e−24
	61	31.1	> 13312	14.30	
65536	15	16.3	> 13312	30.20	
	34	35.3	> 13312	1.33 h	
	61	62.2	> 13312	1.67 h	
131072	15	32.5	> 13312	3.52 h	
	34	70.5	> 13312	9.34 h	
	61	124.4	> 13312	11.68 h	
262144	15	65.0	> 13312	1.0 dias	
	34	141.0	> 13312	2.7 dias	
	61	248.9	> 13312	3.4 dias	
524288	15	130.0	> 13312	7.2 dias	
	34	282.0	> 13312	19.1 dias	
	61	498.1	> 13312	23.8 dias	
1048576	15	260.0	> 13312	50.4 dias	
	34	564.0	> 13312	133.5 dias	
	61	995.9	> 13312	166.9 dias	

Tabela 3.2: Resultados obtidos na execução da rotina ma57 sobre instâncias de dimensão e esparsidade variadas.

As entradas abaixo do traço horizontal na Tabela 3.2 correspondem às instâncias para as quais a memória requerida pelo método foi maior que 13GB. Para essas instâncias apenas a primeira etapa do método foi executada, o tempo exibido é uma estimativa do tempo de execução. A estimativa é feita observando que, com algumas exceções, o tempo de execução é aproximadamente 7 vezes maior quando dobramos a dimensão n e mantemos p fixo, ou $\text{tempo}_{2n,\,p} \approx 7\,\text{tempo}_{n,\,p}$.

Outro obstáculo para execução da rotina ma57 é o requerimento de memória. No caso da instância $(32768, 34)$, por exemplo, o método requer aproximadamente é 700 vezes a quantidade de memória ocupada pela matriz A. Além dessa limitação a implementação testada possui uma limitação relacionada a quantidade de memória que a fatoração da matriz pode ocupar. A fatoração é armazenada em um vetor de números reais de 8 bytes e um de inteiros de 4 bytes. Os parâmetros

utilizados para controlar o tamanho de ambos os vetores são inteiros de 32 bits, o que limita o tamanho desses vetores a $2^{31} - 1$ elementos. Assim a quantidade máxima de memória utilizada pela fatoração é de $(2^{31} - 1)(8 + 4)$ bytes, ou aproximadamente 24 GB. Os dados da Tabela 3.2 sugerem que esse limite seria atingido já nas instâncias $(65536, p)$.

Métodos iterativos

Na Tabela 4.2 são listados os métodos iterativos implementados e os rótulos utilizados para identificar cada uma das implementações. Os métodos de busca em bloco de coordenada (CM, CD, UD e BD) foram executados com blocos de tamanho 1, 2, 16 e 64. As execuções com os diferentes tamanhos de blocos são identificadas pela concatenação do tamanho de bloco ao rótulo da implementação (ex: CM1 e CM32 para minimização cíclica com blocos unitários e blocos de 32 coordenadas, respectivamente).

Método	Rótulo
Minimização cíclica	CM
Descida cíclica	CD
Descida aleatória uniforme	UD
Descida aleatória enviesada	BD
Máxima descida com Armijo	SA
Máxima descida com minimização exata	SM
Máxima descida espectral	SS

Tabela 3.3: Métodos implementados.

3.2.2 Resultados

Nas tabelas a seguir são apresentados os resultados das execuções. São 12 tabelas, uma cara cada instância. Cada tabela exibe o resultado da execução de 7 métodos, identificados na coluna m pelos rótulos da Tabela 4.2. A coluna Os 4 métodos de busca em coordenada recebem como parâmetro o tamanho do bloco de variáveis, foram utilizados os valores $\{1,\ 2,\ 16,\ 64\}$, identificado na coluna b. A execução do método xx com tamanho de bloco y sera denotada por xxy, por exemplo, o método CM com bloco unitário é identificado por $cm1$.

Alguns resultados experimentais podem ser previstos observando as informações da Tabela 3.1: a execução $cm1$ e $cd1$ são idênticas, portanto devem exibir resultados muito próximos; se para um valor de tamanho de bloco b os valores $\max L_i$ e $\min L_i$ forem próximos então as execuções bdb e udb também devem conter resultados parecidos.

As execuções foram realizadas por uma quantidade fixa de tempo. O tempo de execução foi escolhido arbitrariamente sendo $N/2048$, onde N é a dimensão da matriz correspondente à instância. Os tempos resultantes são dados na Tabela 3.4.

N	tempo
131072	64
262144	128
524288	256
1048576	512

Tabela 3.4: Tempo de execução.

Análise

O resultado das execuções pode ser resumido da seguinte maneira: o método SS (máxima descida espectral) obteve os melhores resultados em todas as execuções. Em segundo lugar, com resultados

similares, ficam os métodos gradientes *SA* e *SM* e os métodos de busca em coordenada com blocos de tamanho 16 e 64. Esse resumo pode ser facilmente observado na Figura 3.1. A barra correspondente ao método *SS* é aproximadamente duas ordens de grandeza menor que a segunda menor barra.

Podemos observar também que o método *CM* (que como já mencionado é o mais comum na literatura) obteve os piores resultados. Além disso, o aumento dos blocos no método *CM* não resulta em melhor performance, o que pode ser explicado pelo custo de iteração $O(p + b^2)$.

Os método *CD* e *UD* diferem somente na ordem em que as coordenadas são alteradas. Comparando esses dois métodos podemos observar que o método *UD* se sai pior ou igual ao método *CD* nas instâncias mais esparsas ($p = 16$). Porém a ordenação aleatória do método *UD* oferece resultados significativamente melhores para as instâncias mais densas ($p = 64$).

Não é segredo que o acesso aleatório à memória é péssimo para performance devido ao *erro de cache* (*cache miss*). De fato, medições mostraram que enquanto o método *CD* resultam em 30% de *cache miss*, enquanto que em execução equivalente do método *UD* observou-se quase 70% de *cache miss*. A melhora na performance com o aumento dos tamanhos de blocos também pode ser atribuído à melhor utilização de *cache*.

Os métodos *UD* e *BD* apresentaram resultados muito próximos. Pouco pode ser feito em relação ao acesso do vetor de variáveis no caso do método *UD* para melhor aproveitamento de memória *cache*. Porém o método *BD* nos permite determinar quais as componentes do vetor de variáveis serão mais acessados. Essa informação pode ser utilizada para reordenar o vetor de variáveis de maneira a tirar maior proveito da memória cache ou qualquer hierarquia de memória envolvida.

Na Figura observamos o valor da função objetivo (eixo y) em relação à quantidade de iterações (eixo x). Observamos que, por iteração, os métodos de busca em coordenada e os métodos gradientes (exceto *SS*) possuem comportamento semelhante. Essa informação sugere que dificilmente o método *CM* será capaz de superar os demais, mesmo melhorando o custo por iteração.

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	2079	64.0	1.78e+00	210.44
	2	2546	64.0	1.32e+00	155.37
	16	3340	64.0	8.83e−01	103.76
	64	1728	64.0	2.35e+00	277.85
cd	1	2360	64.0	1.48e+00	174.02
	2	2969	64.0	8.68e−01	101.97
	16	3975	64.0	4.47e−01	52.04
	64	3867	64.0	4.33e−01	50.34
ud	1	1082	64.0	3.24e+00	383.03
	2	1775	64.0	1.46e+00	171.63
	16	3553	64.0	5.14e−01	59.98
	64	3748	64.0	4.46e−01	51.87
bd	1	970	64.0	3.86e+00	456.60
	2	1653	64.0	1.68e+00	198.05
	16	3514	64.0	5.47e−01	63.86
	64	3745	64.0	4.85e−01	56.55
ss		2205	64.0	8.43e−03	0.00
sa		1238	64.0	3.63e−01	42.09
sm		2347	64.0	5.29e−01	61.78

(a) $p = 16$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	1456	64.0	2.34e+01	791.21
	2	1752	64.0	1.75e+01	589.27
	16	1788	64.0	1.69e+01	570.46
	64	1296	64.0	2.82e+01	952.07
cd	1	1671	64.0	1.88e+01	635.45
	2	1988	64.0	1.27e+01	428.95
	16	2226	64.0	8.91e+00	300.11
	64	2244	64.0	8.23e+00	277.12
ud	1	918	64.1	1.59e+01	537.61
	2	1345	64.0	8.99e+00	302.80
	16	2070	64.0	4.04e+00	135.65
	64	2210	64.0	3.45e+00	115.58
bd	1	843	64.1	1.75e+01	590.12
	2	1275	64.0	9.72e+00	327.51
	16	2058	64.0	4.47e+00	149.93
	64	2293	64.0	3.95e+00	132.35
ss		1326	64.2	2.96e−02	0.00
sa		540	64.0	3.35e+00	112.08
sm		1348	64.0	5.65e+00	189.82

(b) $p = 32$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	935	64.0	4.11e+02	2350.15
	2	1084	64.0	3.18e+02	1817.29
	16	984	64.0	3.76e+02	2149.47
	64	876	64.0	4.61e+02	2636.13
cd	1	1075	64.0	3.23e+02	1843.62
	2	1186	64.0	2.45e+02	1396.43
	16	1182	64.0	2.07e+02	1179.80
	64	1213	64.0	1.85e+02	1056.53
ud	1	694	64.0	9.61e+01	548.33
	2	914	64.0	6.04e+01	343.94
	16	1126	64.0	4.08e+01	232.03
	64	1204	64.0	3.10e+01	175.92
bd	1	656	64.1	1.02e+02	580.51
	2	881	64.0	6.18e+01	352.19
	16	1111	64.0	4.69e+01	267.04
	64	1206	64.0	3.92e+01	222.94
ss		717	64.0	1.75e−01	0.00
sa		227	64.0	3.82e+01	217.49
sm		724	64.0	7.50e+01	427.74

(c) $p = 64$

Tabela 3.5: Resutado da execução para instâncias de dimensão $N = 131072$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	2091	128.0	3.55e+00	128.80
	2	2385	128.0	2.91e+00	105.54
	16	3146	128.0	1.92e+00	69.35
	64	1449	128.1	6.17e+00	224.79
cd	1	2151	128.0	3.40e+00	123.40
	2	2646	128.0	2.08e+00	74.95
	16	3245	128.0	1.23e+00	44.00
	64	3018	128.0	1.28e+00	45.72
ud	1	850	128.1	9.54e+00	347.86
	2	1385	128.0	4.15e+00	150.70
	16	2804	128.0	1.46e+00	52.40
	64	2895	128.0	1.33e+00	47.82
bd	1	779	128.1	1.07e+01	389.15
	2	1283	128.0	4.78e+00	173.80
	16	2716	128.0	1.55e+00	55.63
	64	2910	128.0	1.37e+00	48.99
ss		1666	128.1	2.73e−02	0.00
sa		924	128.2	1.17e+00	41.75
sm		1734	128.1	1.64e+00	59.11

(a) $p = 16$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	1337	128.1	5.49e+01	378.81
	2	1553	128.1	4.36e+01	301.03
	16	1533	128.0	4.45e+01	307.09
	64	1020	128.0	8.32e+01	574.70
cd	1	1525	128.0	4.49e+01	309.55
	2	1734	128.0	3.23e+01	222.43
	16	1737	128.1	2.64e+01	181.94
	64	1773	128.0	2.38e+01	163.89
ud	1	727	128.1	4.44e+01	306.72
	2	1081	128.0	2.47e+01	169.75
	16	1592	128.1	1.29e+01	88.48
	64	1705	128.0	1.15e+01	78.59
bd	1	669	128.1	4.96e+01	342.11
	2	1054	128.1	2.41e+01	165.51
	16	1941	128.0	1.00e+01	68.45
	64	1710	128.1	1.15e+01	78.95
ss		1032	128.1	1.44e−01	0.00
sa		467	128.1	8.56e+00	58.29
sm		1001	128.1	1.77e+01	121.30

(b) $p = 32$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	838	128.0	1.03e+03	1239.28
	2	962	128.0	8.04e+02	971.11
	16	993	128.0	7.61e+02	919.00
	64	689	128.1	1.46e+03	1768.97
cd	1	940	128.1	8.37e+02	1011.19
	2	1016	128.0	6.49e+02	783.15
	16	948	128.1	6.05e+02	730.27
	64	963	128.0	5.49e+02	662.47
ud	1	560	128.1	2.60e+02	313.03
	2	736	128.1	1.70e+02	204.35
	16	1013	128.1	9.84e+01	118.03
	64	931	128.0	1.12e+02	134.31
bd	1	537	128.2	2.78e+02	335.53
	2	730	128.1	1.69e+02	203.69
	16	882	128.0	1.25e+02	150.44
	64	945	128.1	1.18e+02	142.24
ss		545	128.0	8.27e−01	0.00
sa		180	128.9	1.10e+02	131.88
sm		574	128.1	2.10e+02	252.86

(c) $p = 64$

Tabela 3.6: Resutado da execução para instâncias de dimensão $N = 262144$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	1592	256.1	1.06e+01	75.12
	2	1820	256.0	8.62e+00	60.96
	16	2350	256.1	5.82e+00	40.81
	64	1340	256.1	1.38e+01	98.21
cd	1	1728	256.1	9.34e+00	66.11
	2	2060	256.0	5.96e+00	41.85
	16	2455	256.1	3.69e+00	25.50
	64	2320	256.0	3.74e+00	25.88
ud	1	728	256.2	2.38e+01	170.31
	2	1092	256.1	1.23e+01	87.26
	16	2170	256.0	4.23e+00	29.42
	64	2210	256.1	4.03e+00	27.94
bd	1	673	256.2	2.63e+01	187.95
	2	1029	256.0	1.35e+01	96.04
	16	2063	256.0	4.63e+00	32.28
	64	2190	256.0	4.36e+00	30.31
ss		1065	256.5	1.39e−01	0.00
sa		517	256.1	5.33e+00	37.29
sm		1059	256.2	6.81e+00	47.96

(a) $p = 16$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	1067	256.1	1.54e+02	238.55
	2	1186	256.2	1.31e+02	202.98
	16	1221	256.2	1.26e+02	194.18
	64	800	256.2	2.40e+02	371.64
cd	1	1151	256.1	1.37e+02	212.47
	2	1291	256.1	1.00e+02	155.04
	16	1335	256.0	7.79e+01	120.01
	64	1451	256.0	6.37e+01	98.02
ud	1	605	256.4	1.16e+02	180.03
	2	819	256.2	7.14e+01	109.89
	16	1215	256.1	3.89e+01	59.40
	64	1278	256.1	3.32e+01	50.62
bd	1	532	256.3	1.42e+02	219.09
	2	783	256.2	7.73e+01	119.12
	16	1199	256.1	3.99e+01	61.04
	64	1262	256.1	3.76e+01	57.39
ss		616	256.1	6.44e−01	0.00
sa		225	260.5	5.20e+01	79.78
sm		645	256.3	6.85e+01	105.52

(b) $p = 32$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	561	256.1	4.15e+03	448.03
	2	624	256.3	3.42e+03	369.10
	16	620	256.2	3.46e+03	373.43
	64	558	256.1	4.19e+03	452.39
cd	1	706	256.0	2.74e+03	295.46
	2	738	256.2	2.21e+03	237.39
	16	695	256.2	1.97e+03	212.45
	64	702	256.2	1.79e+03	192.83
ud	1	432	256.1	7.67e+02	81.89
	2	546	256.4	5.29e+02	56.22
	16	659	256.1	3.85e+02	40.64
	64	690	256.3	3.24e+02	34.01
bd	1	422	256.0	7.93e+02	84.68
	2	561	256.3	4.90e+02	51.99
	16	658	256.4	3.84e+02	40.48
	64	688	256.0	3.72e+02	39.25
ss		258	256.2	9.25e+00	0.00
sa		90	258.3	7.70e+02	82.25
sm		282	256.3	1.19e+03	127.77

(c) $p = 64$

Tabela 3.7: Resutado da execução para instâncias de dimensão $N = 524288$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	1128	512.2	3.63e+01	56.56
	2	1240	512.3	3.15e+01	48.86
	16	1423	512.4	2.55e+01	39.46
	64	1142	512.0	3.57e+01	55.50
cd	1	1207	512.4	3.28e+01	50.94
	2	1306	512.0	2.43e+01	37.49
	16	1459	512.1	1.65e+01	25.11
	64	1422	512.2	1.59e+01	24.22
ud	1	649	512.2	5.62e+01	88.05
	2	837	512.4	3.67e+01	57.10
	16	1339	512.2	1.76e+01	26.85
	64	1382	512.3	1.70e+01	25.88
bd	1	613	512.3	6.17e+01	96.78
	2	852	512.2	3.56e+01	55.38
	16	1328	512.4	1.81e+01	27.69
	64	1387	512.4	1.69e+01	25.81
ss		588	512.5	6.31e−01	0.00
sa		309	515.8	2.36e+01	36.31
sm		600	512.3	3.22e+01	49.96

(a) $p = 16$

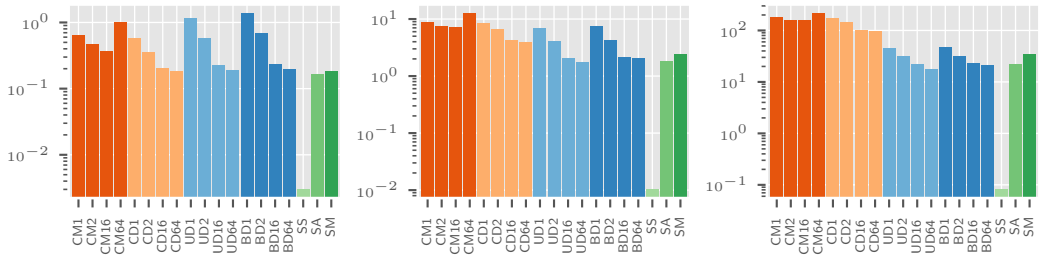
m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	668	512.3	6.41e+02	126.99
	2	729	512.2	5.58e+02	110.53
	16	780	512.2	5.02e+02	99.29
	64	608	512.6	7.44e+02	147.53
cd	1	717	512.2	5.73e+02	113.48
	2	733	512.2	4.81e+02	95.06
	16	653	512.2	4.68e+02	92.46
	64	653	512.7	4.33e+02	85.49
ud	1	484	521.6	3.27e+02	64.26
	2	599	512.2	2.30e+02	44.93
	16	755	512.2	1.56e+02	30.14
	64	783	512.1	1.49e+02	28.77
bd	1	469	523.3	3.41e+02	67.03
	2	583	512.9	2.42e+02	47.43
	16	763	512.7	1.56e+02	30.23
	64	767	512.4	1.54e+02	29.72
ss		344	515.8	5.01e+00	0.00
sa		130	527.5	2.42e+02	47.25
sm		362	560.3	3.23e+02	63.46

(b) $p = 32$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
cm	1	395	549.0	1.61e+04	558.03
	2	417	558.3	1.46e+04	505.88
	16	406	553.3	1.54e+04	530.99
	64	362	529.5	1.89e+04	652.31
cd	1	397	523.8	1.60e+04	552.97
	2	406	527.4	1.33e+04	459.87
	16	349	519.7	1.38e+04	475.06
	64	428	535.6	8.65e+03	298.46
ud	1	331	534.1	2.26e+03	77.21
	2	388	545.6	1.74e+03	59.35
	16	423	532.9	1.51e+03	51.38
	64	424	534.0	1.45e+03	49.06
bd	1	309	516.1	2.52e+03	86.34
	2	375	537.9	1.87e+03	63.61
	16	423	533.6	1.44e+03	48.83
	64	426	535.2	1.46e+03	49.43
ss		160	516.0	2.89e+01	0.00
sa		50	515.5	3.65e+03	125.31
sm		160	514.7	5.34e+03	183.80

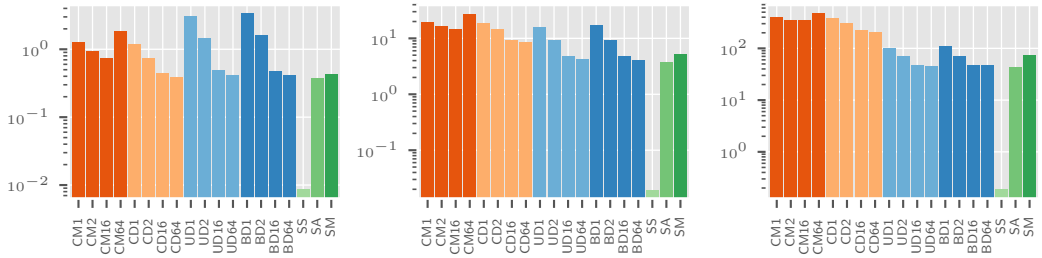
(c) $p = 64$

Tabela 3.8: Resutado da execução para instâncias de dimensão $N = 1048576$



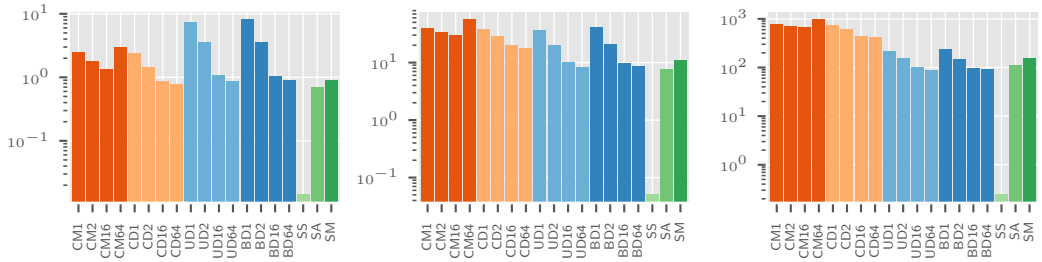
$p = 16$ $p = 32$ $p = 64$

(a) $N = 131072 \quad t = 32s$



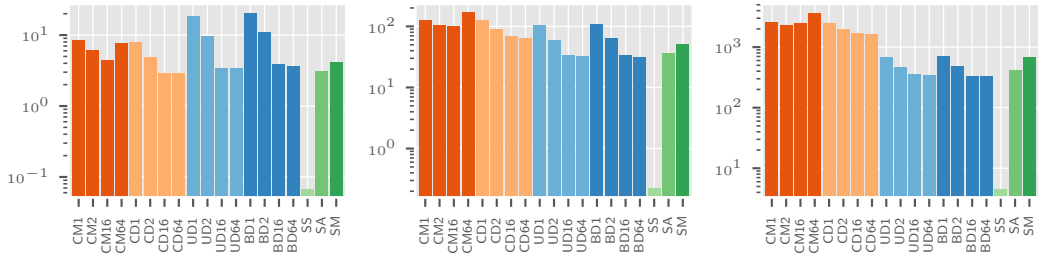
$p = 16$ $p = 32$ $p = 64$

(b) $N = 262144 \quad t = 64s$



$p = 16$ $p = 32$ $p = 64$

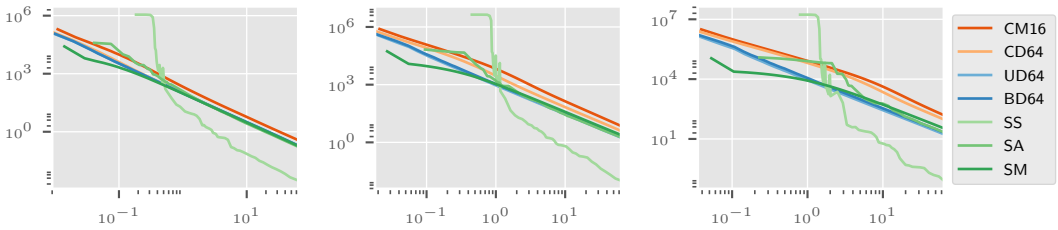
(c) $N = 524288 \quad t = 128s$



$p = 16$ $p = 32$ $p = 64$

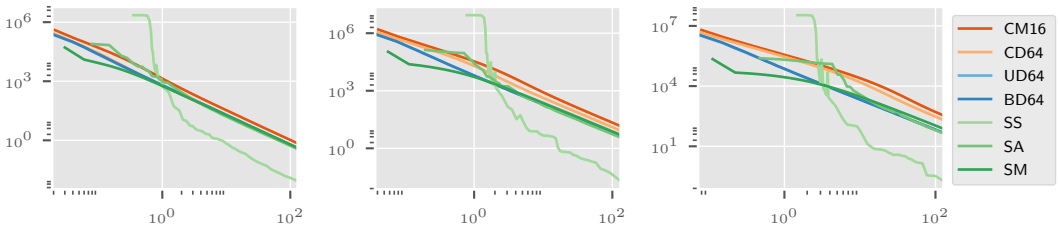
(d) $N = 1048576 \quad t = 256s$

Figura 3.1: Comparação do valor da função objetivo.



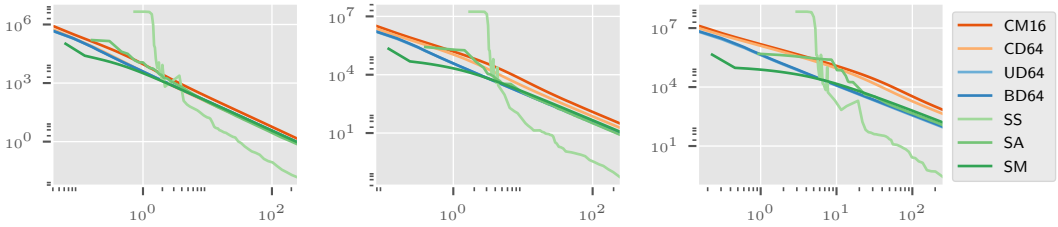
$p = 16$ $p = 32$ $p = 64$

(a) $N = 131072$



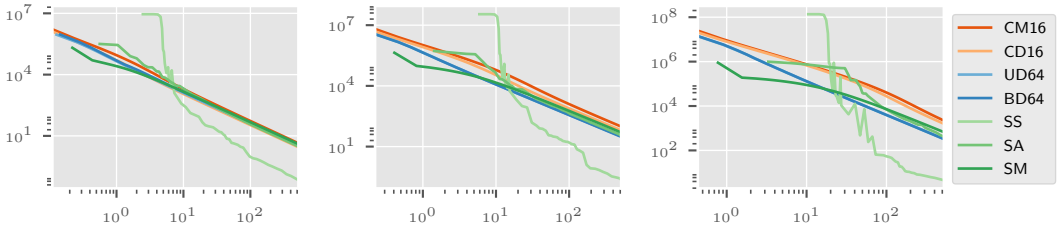
$p = 16$ $p = 32$ $p = 64$

(b) $N = 262144$



$p = 16$ $p = 32$ $p = 64$

(c) $N = 524288$



$p = 16$ $p = 32$ $p = 64$

(d) $N = 1048576$

Figura 3.2: Valor da função objetivo em relação à iteração para as primeiras cem iterações

3.3 Instâncias Sobredeterminadas

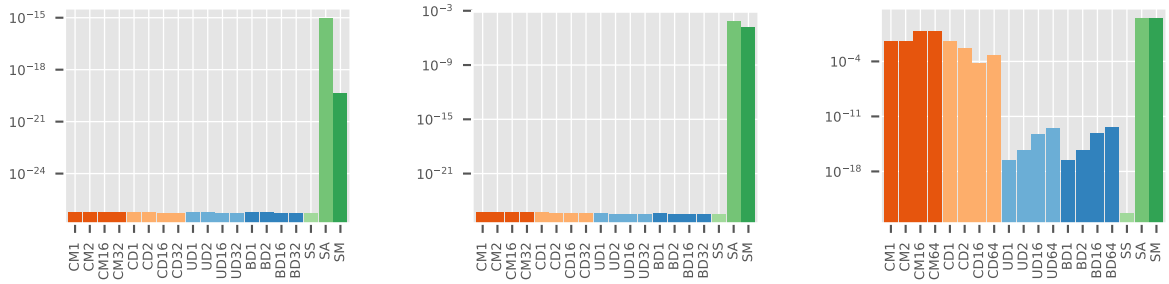
Além das instâncias com matrizes quadradas e problemas estritamente convexos da sessão anterior também foram executados experimentos sobre instâncias sobredeterminadas, isto é, matrizes em que a quantidade de linhas é maior que a quantidade de colunas. Em um primeiro conjunto de instâncias os valores escolhidos para $(N\, p)$ foram os mesmo descritos na Seção 3.2.1,

$$(N,\, p) \in \{131072,\, 262144,\, 524288,\, 1048576\} \times \{16,\, 32,\, 64\},$$

sendo $m = 2N$, onde m é a quantidade de linhas da matriz A. Para esse conjunto de instâncias os resultados são resumidos pela Figura 3.3. Nessa figura observamos que para valores baixos de p, exceto pelos métodos *SM* e *SA*, todas execuções apresentaram capacidade semelhante de minimização da função objetivo. Para obtermos resultados mais interessantes, um novo conjunto de instâncias foi gerado. Nesse novo conjunto de instâncias a média de elementos não nulos por coluna é $p \in \{64,\, 128,\, 256\}$. No total foram geradas doze instâncias, sobre as quais foram executados dezenove métodos, totalizando 228 execuções. Novamente, cada método foi executada por uma quantidade fixa de tempo, proporcional à quantidade de variáveis da instância. Esses tempos são listados na Tabela 3.9.

N	tempo
131072	32
262144	54
524288	128
1048576	256

Tabela 3.9: Tempo de execução.



(a) $p = 16$ (b) $p = 32$ (c) $p = 64$

Figura 3.3: $N = 131072$, $m = 2N$, 32s de execução

A Tabela contém informações sobre os blocos obtidos em cada uma das instâncias. Com base nessas informações podemos prever que os métodos *UD* e *BD* apresentarão resultados semelhantes para $b \in \{16,\, 64\}$, pois nesses casos a diferença entre L_{min} e L_{max} é pequena. Enquanto que os métodos *CD* e *CM* terão performance semelhante para $b \in \{1,\, 2\}$, pois a matriz Hessiana em correspondente a cada bloco é diagonal na maioria dos casos.

As Tabelas 3.11, 3.12, 3.13 e 3.14 apresentam os resultados de cada execução, esses dados são apresentados com objetivo de complementar as informações ilustradas figuras seguintes.

p	b	blocos	diag.	$\min L_i$	$\max L_i$
64	1	131072	100.0%	11.6	32.5
	2	65536	98.5%	15.2	32.5
	16	8192	15.5%	22.0	32.5
	64	2048	0.0%	24.1	32.5
128	1	131072	100.0%	28.6	60.8
	2	65536	93.9%	33.7	60.8
	16	8192	0.0%	43.5	60.8
	64	2048	0.0%	46.7	60.8
256	1	131072	100.0%	63.7	106.0
	2	65536	77.9%	72.3	106.0
	16	8192	0.0%	85.5	106.0
	64	2048	0.0%	91.8	106.1

(a) $N = 131072$

p	b	blocos	diag.	$\min L_i$	$\max L_i$
64	1	262144	100.0%	11.8	32.6
	2	131072	99.2%	14.6	32.6
	16	16384	39.7%	21.8	32.6
	64	4096	0.0%	24.2	32.6
128	1	262144	100.0%	27.1	59.5
	2	131072	96.9%	32.9	59.5
	16	16384	2.3%	41.7	59.5
	64	4096	0.0%	46.0	59.5
256	1	262144	100.0%	64.5	106.7
	2	131072	88.2%	73.0	106.7
	16	16384	0.0%	85.9	106.7
	64	4096	0.0%	90.7	106.7

(b) $N = 262144$

p	b	blocos	diag.	$\min L_i$	$\max L_i$
64	1	524288	100.0%	11.0	32.2
	2	262144	99.6%	14.8	32.2
	16	32768	62.4%	21.4	32.2
	64	8192	0.0%	24.1	32.2
128	1	524288	100.0%	28.1	58.5
	2	262144	98.4%	33.2	58.5
	16	32768	15.2%	43.0	58.5
	64	8192	0.0%	46.5	58.5
256	1	524288	100.0%	62.7	107.7
	2	262144	94.0%	71.9	107.7
	16	32768	0.0%	85.5	107.7
	64	8192	0.0%	90.8	107.7

(c) $N = 524288$

p	b	blocos	diag.	$\min L_i$	$\max L_i$
64	1	1048576	100.0%	10.0	33.6
	2	524288	99.8%	14.4	33.6
	16	65536	79.1%	21.4	33.6
	64	16384	2.1%	23.7	33.6
128	1	1048576	100.0%	25.8	59.8
	2	524288	99.2%	33.2	59.8
	16	65536	39.0%	42.2	59.8
	64	16384	0.0%	46.4	59.8
256	1	1048576	100.0%	59.9	108.7
	2	524288	96.9%	72.0	108.7
	16	65536	2.3%	85.8	108.7
	64	16384	0.0%	90.7	108.8

(d) $N = 1048576$

Tabela 3.10: Informações sobre os blocos obtidos em cada uma das instâncias de acordo com o tamanho de bloco escolhido

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	260	36.1	3.48e−02	8.46e+21
	2	260	35.1	3.47e−02	8.45e+21
	16	214	32.0	6.09e−01	1.48e+23
	64	214	34.1	5.79e−01	1.41e+23
CD	1	260	33.8	3.48e−02	8.46e+21
	2	260	35.5	3.97e−03	9.65e+20
	16	260	37.2	4.81e−05	1.17e+19
	64	214	34.4	5.29e−04	1.29e+20
UD	1	260	35.7	2.71e−17	6.59e+06
	2	260	35.2	4.95e−16	1.20e+08
	16	260	34.4	5.58e−14	1.36e+10
	64	260	37.2	3.34e−13	8.12e+10
BD	1	260	36.3	2.23e−17	5.41e+06
	2	260	35.6	4.25e−16	1.03e+08
	16	260	34.7	6.45e−14	1.57e+10
	64	260	37.3	4.51e−13	1.10e+11
SM		176	38.5	3.35e+01	8.14e+24
SA		52	36.3	3.00e+01	7.30e+24
SS		214	33.7	4.11e−24	0.00e+00

(a) $p = 64$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	145	35.3	6.79e+04	1.99e+16
	2	145	36.3	6.79e+04	1.99e+16
	16	119	33.2	1.04e+05	3.05e+16
	64	119	34.4	1.02e+05	3.00e+16
CD	1	145	35.1	6.79e+04	1.99e+16
	2	145	36.2	5.45e+04	1.60e+16
	16	119	32.1	6.01e+04	1.76e+16
	64	119	33.3	5.10e+04	1.50e+16
UD	1	145	37.9	7.48e−08	2.20e+04
	2	145	37.0	2.50e−07	7.33e+04
	16	119	32.6	1.03e−04	3.04e+07
	64	119	33.7	1.41e−04	4.15e+07
BD	1	119	33.1	8.15e−06	2.39e+06
	2	145	39.0	2.07e−07	6.09e+04
	16	119	32.8	1.19e−04	3.50e+07
	64	119	33.8	2.17e−04	6.36e+07
SM		79	34.6	2.81e+03	8.25e+14
SA		21	36.2	5.61e+03	1.65e+15
SS		119	36.0	3.41e−12	0.00e+00

(b) $p = 128$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	79	37.2	1.50e+06	6.40e+07
	2	79	38.8	1.50e+06	6.40e+07
	16	64	35.4	1.75e+06	7.44e+07
	64	64	36.1	1.72e+06	7.35e+07
CD	1	79	38.8	1.50e+06	6.40e+07
	2	79	38.9	1.43e+06	6.11e+07
	16	64	34.0	1.54e+06	6.59e+07
	64	64	33.5	1.50e+06	6.39e+07
UD	1	64	32.2	1.17e+00	4.87e+01
	2	64	32.7	1.53e+00	6.42e+01
	16	64	35.6	2.33e+00	9.84e+01
	64	64	35.7	2.36e+00	9.96e+01
BD	1	64	32.7	9.82e−01	4.09e+01
	2	64	32.7	1.44e+00	6.04e+01
	16	64	34.5	3.23e+00	1.37e+02
	64	64	35.0	3.64e+00	1.54e+02
SM		42	35.1	2.92e+04	1.25e+06
SA		9	37.5	3.09e+05	1.32e+07
SS		52	32.9	2.34e−02	0.00e+00

(c) $p = 256$

Tabela 3.11: Resutado da execução para instâncias de dimensão $N = 131072$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	315	76.7	2.24e−03	3.67e+19
	2	315	76.3	2.24e−03	3.67e+19
	16	260	65.8	6.90e−02	1.13e+21
	64	260	77.3	6.69e−02	1.09e+21
CD	1	315	76.8	2.24e−03	3.67e+19
	2	315	76.1	1.59e−04	2.60e+18
	16	260	65.6	9.44e−05	1.54e+18
	64	260	71.2	1.22e−05	2.00e+17
UD	1	260	74.8	3.64e−17	5.96e+05
	2	260	69.6	7.44e−16	1.22e+07
	16	260	65.7	1.22e−13	1.99e+09
	64	260	72.5	8.91e−13	1.46e+10
BD	1	260	77.0	4.28e−17	7.01e+05
	2	260	73.8	9.24e−16	1.51e+07
	16	260	68.1	1.28e−13	2.09e+09
	64	260	73.1	1.07e−12	1.75e+10
SM		176	76.6	6.62e+01	1.08e+24
SA		52	72.2	5.71e+01	9.33e+23
SS		214	76.5	6.12e−23	0.00e+00

(a) $p = 64$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	145	72.8	1.36e+05	2.56e+12
	2	119	65.1	2.09e+05	3.94e+12
	16	97	68.9	3.02e+05	5.67e+12
	64	97	73.5	3.00e+05	5.64e+12
CD	1	119	64.1	2.09e+05	3.94e+12
	2	119	74.3	1.75e+05	3.29e+12
	16	119	68.9	1.21e+05	2.27e+12
	64	119	69.2	1.02e+05	1.91e+12
UD	1	97	69.0	9.43e−04	1.77e+04
	2	97	67.4	2.23e−03	4.19e+04
	16	97	77.2	7.41e−03	1.39e+05
	64	97	77.7	1.21e−02	2.27e+05
BD	1	97	70.4	9.31e−04	1.75e+04
	2	145	74.9	3.89e−07	6.31e+00
	16	119	70.5	2.34e−04	4.40e+03
	64	119	78.0	4.64e−04	8.73e+03
SM		79	78.5	5.67e+03	1.07e+11
SA		16	65.4	3.29e+04	6.18e+11
SS		97	76.0	5.32e−08	0.00e+00

(b) $p = 128$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	52	68.6	4.07e+06	7.10e+06
	2	64	78.3	3.51e+06	6.12e+06
	16	52	76.2	4.07e+06	7.09e+06
	64	42	71.0	4.69e+06	8.17e+06
CD	1	64	72.5	3.51e+06	6.12e+06
	2	64	73.7	3.36e+06	5.86e+06
	16	52	76.5	3.61e+06	6.29e+06
	64	42	66.0	4.09e+06	7.12e+06
UD	1	52	64.3	2.19e+01	3.72e+01
	2	52	71.7	3.13e+01	5.35e+01
	16	64	71.2	5.52e+00	8.62e+00
	64	64	69.2	6.50e+00	1.03e+01
BD	1	64	70.1	1.96e+00	2.41e+00
	2	64	72.2	2.64e+00	3.60e+00
	16	64	78.4	6.14e+00	9.70e+00
	64	52	65.2	7.16e+01	1.24e+02
SM		42	78.0	5.82e+04	1.02e+05
SA		7	69.3	6.83e+05	1.19e+06
SS		52	72.4	5.74e−01	0.00e+00

(c) $p = 256$

Tabela 3.12: Resutado da execução para instâncias de dimensão $N = 262144$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	176	153.2	2.75e+01	4.29e+17
	2	176	155.1	2.75e+01	4.29e+17
	16	145	138.6	1.89e+02	2.96e+18
	64	145	136.2	1.88e+02	2.93e+18
CD	1	176	152.1	2.75e+01	4.29e+17
	2	176	154.0	6.35e+00	9.93e+16
	16	145	132.6	5.12e+00	8.00e+16
	64	145	143.1	1.66e+00	2.59e+16
UD	1	176	145.8	2.85e−10	4.45e+06
	2	176	138.0	2.24e−09	3.50e+07
	16	176	137.2	6.53e−08	1.02e+09
	64	176	144.5	1.98e−07	3.10e+09
BD	1	176	149.8	2.74e−10	4.27e+06
	2	176	140.3	2.20e−09	3.44e+07
	16	176	136.3	5.67e−08	8.85e+08
	64	176	145.1	2.24e−07	3.51e+09
SM		97	144.4	9.04e+02	1.41e+19
SA		34	158.4	7.28e+02	1.14e+19
SS		145	155.3	6.40e−17	0.00e+00

(a) $p = 64$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	119	154.4	4.16e+05	2.18e+08
	2	97	131.4	6.02e+05	3.16e+08
	16	97	153.3	6.01e+05	3.15e+08
	64	79	142.0	8.17e+05	4.29e+08
CD	1	97	129.5	6.02e+05	3.16e+08
	2	97	131.1	5.19e+05	2.72e+08
	16	97	150.6	3.82e+05	2.00e+08
	64	79	130.4	5.03e+05	2.64e+08
UD	1	97	144.0	1.91e−03	0.00e+00
	2	97	137.6	4.09e−03	1.14e+00
	16	97	150.2	1.67e−02	7.76e+00
	64	79	130.3	4.33e−01	2.26e+02
BD	1	97	145.3	1.93e−03	1.25e−02
	2	97	138.4	3.96e−03	1.08e+00
	16	97	150.9	1.46e−02	6.69e+00
	64	79	133.6	5.09e−01	2.66e+02
SM		52	149.5	2.23e+04	1.17e+07
SA		12	142.4	4.86e+05	2.55e+08
SS		64	134.3	4.75e−02	2.40e+01

(b) $p = 128$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	52	137.7	8.10e+06	1.94e+05
	2	52	150.0	8.10e+06	1.94e+05
	16	42	135.4	9.39e+06	2.25e+05
	64	42	155.4	9.36e+06	2.24e+05
CD	1	52	138.8	8.10e+06	1.94e+05
	2	52	148.4	7.77e+06	1.86e+05
	16	42	134.9	8.37e+06	2.00e+05
	64	42	150.0	8.11e+06	1.94e+05
UD	1	52	144.1	4.38e+01	4.67e−02
	2	52	152.6	5.60e+01	3.38e−01
	16	42	133.0	6.86e+02	1.54e+01
	64	42	148.5	6.60e+02	1.48e+01
BD	1	52	145.1	4.18e+01	0.00e+00
	2	52	153.3	5.72e+01	3.67e−01
	16	42	133.0	5.92e+02	1.32e+01
	64	42	150.4	8.19e+02	1.86e+01
SM		27	150.7	1.63e+05	3.89e+03
SA		5	131.1	1.50e+06	3.60e+04
SS		27	129.4	2.59e+02	5.19e+00

(c) $p = 256$

Tabela 3.13: Resutado da execução para instâncias de dimensão $N = 524288$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	145	294.1	3.79e+02	2.32e+07
	2	145	295.8	3.79e+02	2.32e+07
	16	145	311.6	3.79e+02	2.32e+07
	64	119	291.9	1.91e+03	1.17e+08
CD	1	145	294.3	3.79e+02	2.32e+07
	2	145	295.5	1.15e+02	7.06e+06
	16	145	309.9	1.03e+01	6.30e+05
	64	119	275.5	4.09e+01	2.50e+06
UD	1	119	271.5	1.63e−05	0.00e+00
	2	119	259.7	6.58e−05	3.03e+00
	16	119	261.0	6.22e−04	3.71e+01
	64	119	288.5	1.43e−03	8.67e+01
BD	1	119	274.0	1.74e−05	6.69e−02
	2	119	264.6	6.71e−05	3.10e+00
	16	119	264.0	5.90e−04	3.51e+01
	64	119	288.0	1.45e−03	8.77e+01
SM		52	305.9	7.70e+03	4.71e+08
SA		16	301.6	1.31e+04	8.00e+08
SS		64	257.2	5.72e−03	3.49e+02

(a) $p = 64$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	64	281.7	2.15e+06	1.05e+06
	2	64	270.1	2.15e+06	1.05e+06
	16	64	283.7	2.15e+06	1.05e+06
	64	52	294.6	2.72e+06	1.32e+06
CD	1	64	280.4	2.15e+06	1.05e+06
	2	64	288.2	1.94e+06	9.43e+05
	16	52	260.5	2.08e+06	1.01e+06
	64	42	291.0	2.49e+06	1.21e+06
UD	1	64	266.5	2.06e+00	0.00e+00
	2	64	274.6	3.32e+00	6.14e−01
	16	64	284.7	8.19e+00	2.99e+00
	64	52	295.5	8.56e+01	4.06e+01
BD	1	64	273.0	2.11e+00	2.84e−02
	2	64	276.8	3.39e+00	6.48e−01
	16	64	288.5	7.71e+00	2.75e+00
	64	52	302.7	9.01e+01	4.28e+01
SM		21	256.7	1.26e+05	6.13e+04
SA		5	265.1	1.32e+06	6.41e+05
SS		34	296.2	7.61e+00	2.70e+00

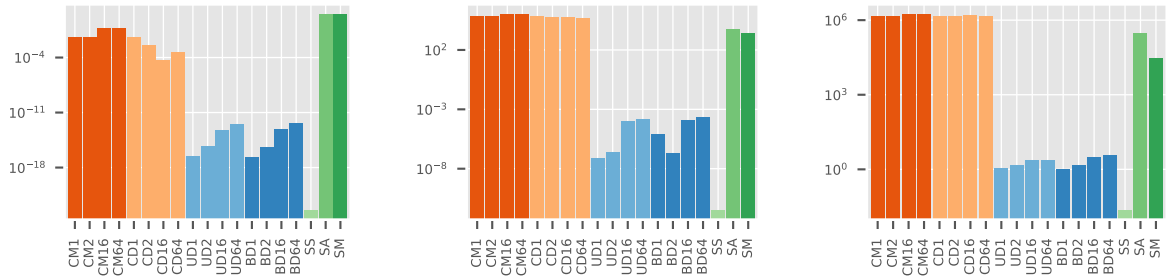
(b) $p = 128$

m	b	it.	t(s)	f	$\frac{(f-f_{\min})}{f_{\min}}$
CM	1	34	300.0	2.16e+07	5.92e+03
	2	34	320.5	2.16e+07	5.92e+03
	16	27	302.3	2.52e+07	6.90e+03
	64	21	299.1	2.97e+07	8.13e+03
CD	1	34	303.6	2.16e+07	5.92e+03
	2	34	311.2	2.08e+07	5.70e+03
	16	27	284.2	2.27e+07	6.21e+03
	64	21	273.6	2.62e+07	7.18e+03
UD	1	34	290.4	3.65e+03	0.00e+00
	2	34	292.2	4.35e+03	1.89e−01
	16	34	316.1	6.46e+03	7.67e−01
	64	21	259.0	1.06e+05	2.80e+01
BD	1	34	281.6	3.74e+03	2.24e−02
	2	34	287.1	4.26e+03	1.66e−01
	16	34	314.9	5.77e+03	5.80e−01
	64	21	257.7	1.17e+05	3.09e+01
SM		12	267.8	4.93e+05	1.34e+02
SA		3	336.9	3.36e+06	9.19e+02
SS		12	275.8	4.81e+04	1.22e+01

(c) $p = 256$

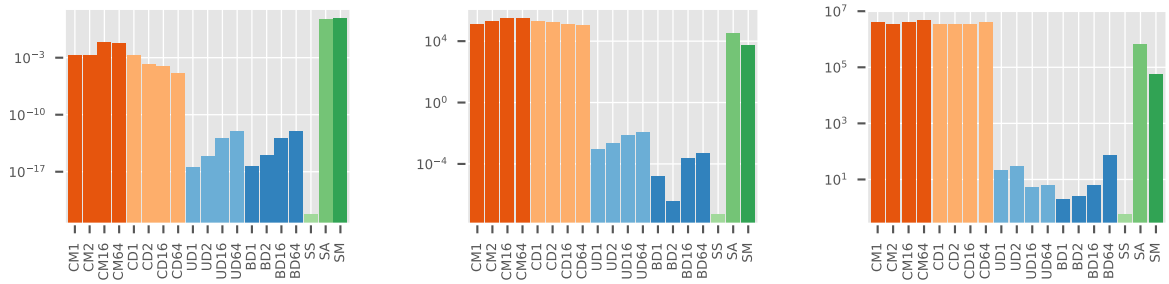
Tabela 3.14: Resutado da execução para instâncias de dimensão $N = 1048576$

As Figuras 3.4, 3.5, 3.6 e 3.7 comparam o valor da função objetivo ao final da execução. Menores valores correspondem à melhores resultados. Para as instâncias sobredeterminadas obtemos diferentes conclusões à aquelas obtidas anteriormente. Agora os métodos *CM*, *CD*, *SM* e *SA* apresentam performance semelhante e ruim quando comparados aos métodos *UD*, *BD* e *SS*. Anteriormente o método *SS* apresentou os melhores resultados em todas as execuções, porém agora os métodos *BD* e *UD* apresentaram resultados competitivos em todos os casos e resultados melhores nas instâncias maiores e mais densas.



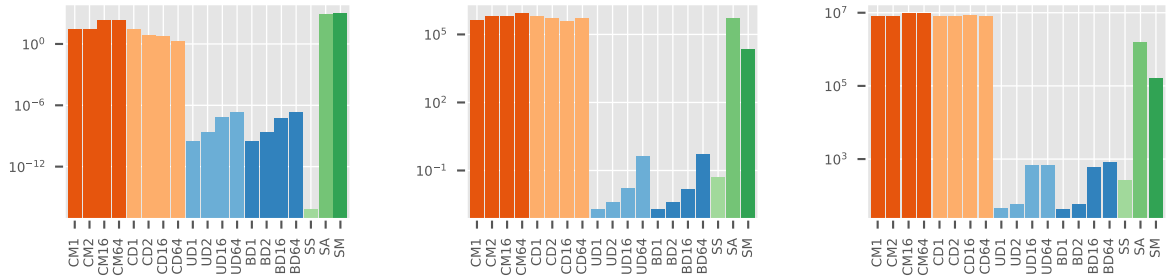
(a) $p = 64$ (b) $p = 128$ (c) $p = 256$

Figura 3.4: $N = 131072$, $m = 2N$, 32s de execução



(a) $p = 64$ (b) $p = 128$ (c) $p = 256$

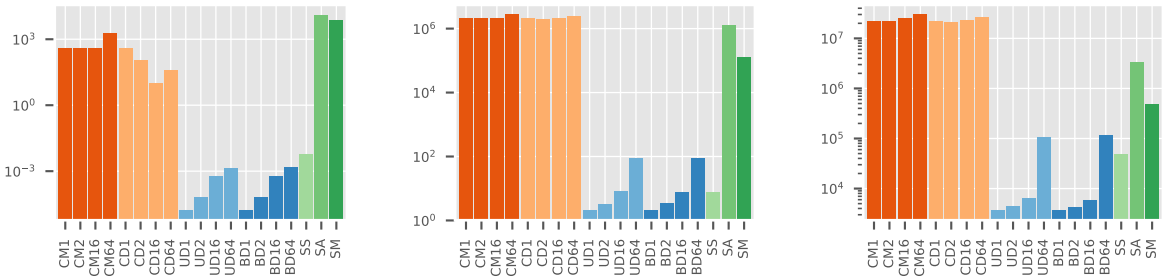
Figura 3.5: $N = 262144$, $m = 2N$, 64s de execução



(a) $p = 64$ (b) $p = 128$ (c) $p = 256$

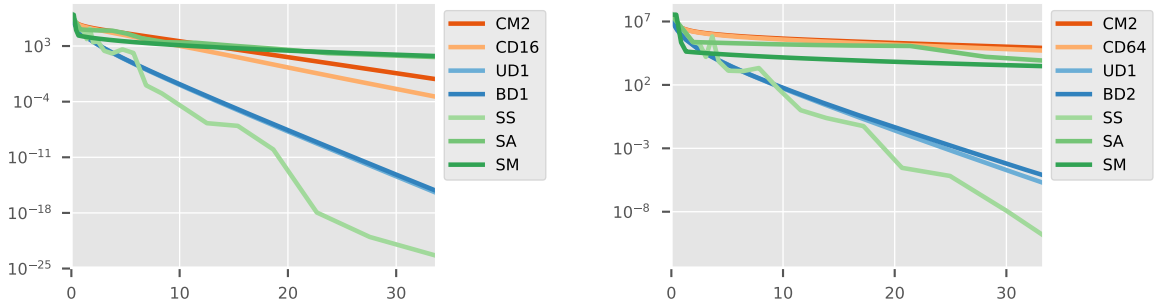
Figura 3.6: $N = 524288$, $m = 2N$, 128s de execução

Poderíamos supor que os resultados seriam diferentes se maior tempo de execução fosse permitido. As Figuras 3.8, 3.9, 3.10 e 3.11 apresentam indícios de esse não seria o caso. Observando essas figuras é sensato supor que os métodos *CM*, *CD*, *SM* e *SA* reduziriam pouco o valor da função objetivo caso mais tempo execução fosse disponibilizado. Já nos métodos *UD*, *BD* e *SS* aparentemente se beneficiariam de maior tempo de execução.

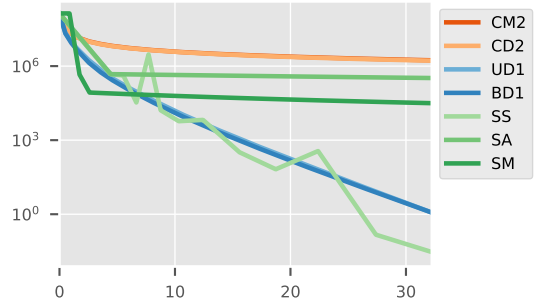


(a) $p = 64$ (b) $p = 128$ (c) $p = 256$

Figura 3.7: $N = 1048576, \quad m = 2N, \quad$ 256s de execução

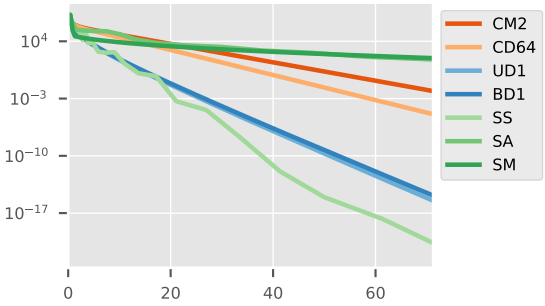


(a) $p = 64$ (b) $p = 128$

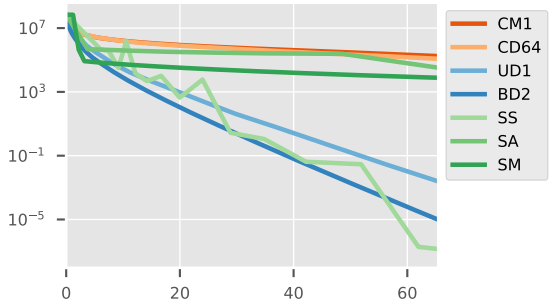


(c) $p = 256$

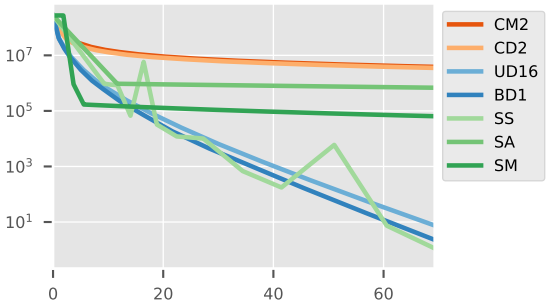
Figura 3.8: $m = 262144, \ N = 131072$



(a) $p = 64$

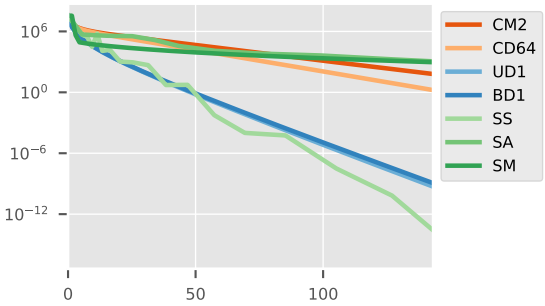


(b) $p = 128$

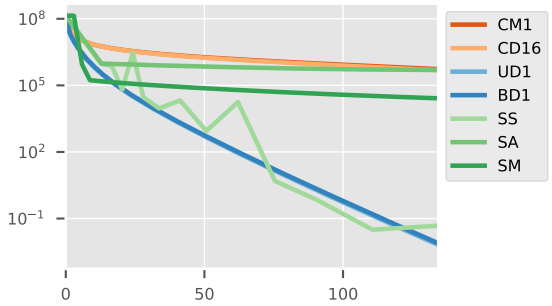


(c) $p = 256$

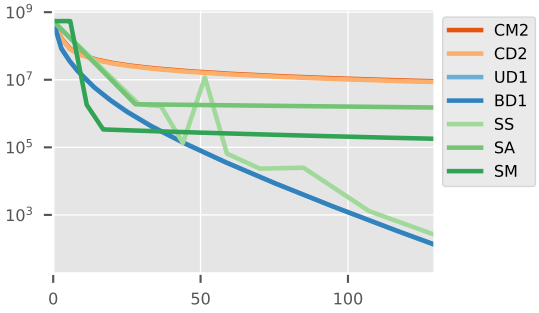
Figura 3.9: $m = 524288,\ N = 262144$



(a) $p = 64$

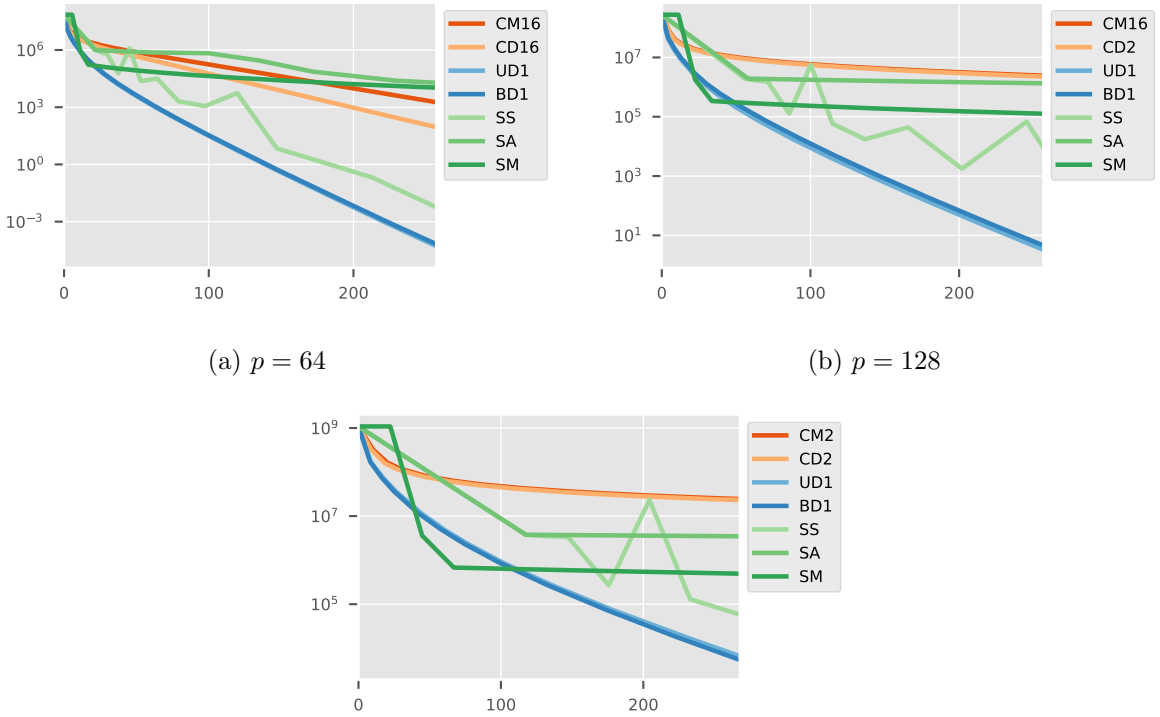


(b) $p = 128$



(c) $p = 256$

Figura 3.10: $m = 1048576,\ N = 524288$



(a) $p = 64$

(b) $p = 128$

(c) $p = 256$

Figura 3.11: $m = 2097152,\ N = 1048576$

Capítulo 4

Pagerank

Em [19] são apresentados resultados experimentais para o problema conhecido como *google problem*. Neste Capítulo atacamos o mesmo problema porém utilizando uma formulação mais próxima à utilizada na prática (ver a observação feita na Seção (4.2.2)). Uma breve descrição para o problema é a seguinte: dado um conjunto de páginas *web* com *links* entre si gostaríamos atribuir a essas páginas um grau de importância e ordena-las de acordo. Um usuário navegando por essas páginas, seguindo links ao acaso, eventualmente encontraria a informação desejada. Durante essa busca ele potencialmente visitaria a mesma página várias vezes. É sensato considerar que as páginas mais visitadas são mais relevantes, pois um elevado número de visitas significa que, direta ou indiretamente, essa página é altamente referenciada pelas demais. Poderíamos então utilizar o número estimado de visitas como o grau de importância da página. Esse valor é um resultado conhecido no estudo de cadeias de Markov e passeios aleatórios em grafos. Na seção seguinte são apresentados alguns conceitos e resultados que serão utilizados para formalizar o problema. As demonstrações para os resultados citados nas próximas seções podem ser encontrados em [18]. Mais informações sobre o problema podem ser encontradas em [6, 23].

4.1 Formulação do Problema

Uma *cadeia de Markov* é um processo estocástico definido por um conjunto de estados $X = \{x_1, \ldots, x_n\}$, e uma matriz de transição $P = \{p_{ij}\} \in \mathbb{R}^{n \times n}$. Em um determinado momento a cadeia de Markov se encontra em exatamente um estado. O valor p_{ij} corresponde à probabilidade de que ocorra uma transição para o estado x_j dado que a cadeia se encontra no estado x_i. Transições ocorrem em intervalos discretos de tempo, $k = 0, 1, \ldots$. Vamos chamar de *distribuição de probabilidades*, ou simplesmente *distribuição*, um vetor $q \in \mathbb{R}^n$ que satisfaça

$$q_i \geq 0, \quad i = 1, \ldots, n, \text{ e} \tag{4.1.1}$$

$$\langle q, e \rangle = 1, \tag{4.1.2}$$

onde e é o vetor em que todas as componentes são iguais a 1. Em um instante k a probabilidade da cadeia se encontrar no estado x_i é dada pelo i-ésimo componente da distribuição q^k. A distribuição q^{k+1} é dada pela expressão

$$q^{k+1} = P^T q^k$$

e por indução

$$q^k = \left[P^T\right]^k q^0,$$

onde q^0 é a distribuição inicial. A distribuição de probabilidade q^* que satisfaz $P^T q^* = q^*$ é denominada *distribuição estacionária* da cadeia, e é de especial interesse para o valor que procuramos. A *existência* e *unicidade* da distribuição estacionária depende de algumas propriedades da cadeia de Markov. Essas propriedades podem ser expressas em termos do grafo subjacente.

Considere o grafo *direcionado* $G = (X, E)$ onde $X = \{x_1, \ldots, x_n\}$ é o conjunto de vértices e $E \subset X \times X$ é o conjunto de arestas. G induz uma cadeia de Markov de estados X e matriz de transição $\bar{E} \in \mathbb{R}^{n\times n}$ definida por

$$\bar{E}_{ji} = \begin{cases} \frac{1}{g_i} & \text{se } (x_i, x_j) \in E, \\ 0 & \text{caso contrário,} \end{cases} \tag{4.1.3}$$

onde g_i é o grau de saída do vértice x_i. Note que

$$\left\langle \bar{E}_i, e \right\rangle = 1, \quad i = 1, \ldots, n. \tag{4.1.4}$$

Um caminho de tamanho m é um sequência de vértices $(v_1, v_2, \cdots, v_m)$ tal que $(v_i, v_{i+1}) \in E,\ i = 1, \ldots, m-1$. Um grafo direcionado é dito *fortemente conexo* se, para quaisquer par de vértices $u, v \in X$ existe um caminho com vértice inicial u e vértice final v. Um grafo é dito *bipartido* se seus vértices podem ser particionados em $X = (X_1, X_2)$ de maneira que $(u, v) \in E$ se e somente se u e v pertencerem a classes distintas da partição (X_1, X_2). Se um grafo é fortemente conexo então a distribuição estacionária da cadeia de Markov induzida é única. Mais ainda, se o grafo *não* é bipartido então

$$\lim_{k\to\infty} \bar{E}^k q^0 = q^* \tag{4.1.5}$$

para qualquer q^0. Finalmente, em uma cadeia de Markov induzida por um grafo fortemente conexo temos

$$\lim_{k\to\infty} \frac{N(x_i, k)}{k} = q_i^*,$$

onde $N(x_i, k)$ é a quantidade de visitas feitas ao estado x_i após k transições.

Assim, considerando o grafo em que os vértices representam um páginas web e as arestas os *links* entre essas páginas podemos obter a quantidade de visitas esperada por um usuário visitando páginas "ao acaso". É importante destacar que *links* de uma página para ela mesma são ignorados, assim

$$\bar{E}_{ii} = 0, \quad i = 1, \ldots, n. \tag{4.1.6}$$

A equação (4.1.5) ainda nos sugere uma maneira de encontrar a distribuição estacionária (ver Seção 4.2.1). Mas existe um problema: o grafo obtido a partir de páginas web é esparso, isto é, possui uma quantidade pequena de arestas. Se n é o número de vértices de um grafo, um grafo completo com o n vértices possui $O(n^2)$ arestas, enquanto que um grafo esparso possui $O(n)$ arestas. É muito provável que este grafo seja bipartido e não seja fortemente conexo. Para contornar este problema seguinte comportamento pode ser adicionado ao nosso usuário: a cada visita o usuário pode ficar "entediado" e recomeçar a sua busca a partir de uma nova página escolhida arbitrariamente. Do ponto de vista do grafo esse comportamento pode ser representado pela adição de novas arestas, tornando o grafo completo. Na cadeia de Markov induzida atribuímos valores menores para as transições associadas a essas novas arestas, preservando assim parte da estrutura original. Algebricamente esse comportamento pode ser expresso pela seguinte alteração da matriz de transição:

$$A = (1 - \mu)\bar{E} + \frac{\mu}{n} S, \tag{4.1.7}$$

onde $S \in \mathbb{R}^{n\times n}$ é a matriz com todos os elementos iguais a 1 e $\mu \in (0, 1)$ é o parâmetro que controla a chance do usuário ficar "entediado" e recomeçar sua busca em outra página. O problema então consiste em resolver sistema linear

$$Ax = x, \tag{4.1.8}$$

ou seja, encontrar um auto-vetor associado ao auto-valor 1 da matriz A.

4.2 Implementações

Nesta Seção descrevemos métodos iterativos para encontrar uma solução aproximada para o sistema (4.1.8). Todas as implementações utilizam o critério de parada

$$\frac{\|Ax - x\|}{\|x\|} \leq \epsilon,$$

onde $\epsilon > 0$ é um parâmetro de execução.

4.2.1 Método das potências

O resultado da equação (4.1.5) nos sugere que uma maneira de se encontrar o auto-vetor desejado é através do *método das potências*. Nesse método a sequência de iterandos é dada pela expressão

$$x^{k+1} = \frac{Ax^k}{\|Ax^k\|}, \tag{4.2.1}$$

onde $\|\cdot\|$ é uma norma qualquer. A normalização impede que o iterando x^k cresça indefinidamente. O resultado (4.1.5) nos garante que o método das potências converge para solução do sistema (4.1.8). É possível demonstrar que para a matriz construída pela expressão (4.1.7) temos

$$\left\|x^{k+1} - x^*\right\|_1 \approx |\lambda_2| \left\|x^k - x^*\right\|_1,$$

onde λ_2 é o auto-valor de A com o segundo maior valor absoluto, e $\lambda_2 \leq 1 - \mu$ (Seção 4 de [6] e referências). Assim, é esperado que o iterando x^k se aproxime rapidamente de x^* quanto maior o valor do parâmetro μ.

Custo computacional

O Algoritmo PMG lista com detalhes as implementação do método das potências aplicada à matriz A. A operação de maior custo é a multiplicação $\bar{E}x^k$ (Passo 2), que requer $O(nz)$ operações.

Algoritmo PMG(Método das potências). Os dados de entrada são: a matriz $A \in \mathbb{R}^{n\times n}$ da forma (4.1.7); o parâmetro $\mu \in (0, 1)$ e a precisão $\epsilon > 0$.

1. [Inicialização]. Inicialize os iterando inicial $x^0 = (1/n)e$, o vetor auxiliar $u \in \mathbb{R}^n$ e contador $k = 0$.

2. [Multiplicação Ax^k]. Faça

$$u \leftarrow (1 - \mu)\bar{E}x^k + \frac{\mu}{n}e.$$

3. [Normalização]. Faça

$$x^{k+1} \leftarrow \frac{u}{\langle e, u\rangle}.$$

4. [Critério de parada]. Se $\left\|u - x^k\right\|_2 \leq \epsilon \left\|x^k\right\|_2$ pare e retorne x^{k+1}. Caso contrário incremente k e volte para o passo 2. ∎

Note que se $x^0 > 0$ então $x^k > 0$ para $k = 0, 1, \ldots$ e portanto $\left\langle e, x^k\right\rangle = \left\|x^k\right\|_1$.

4.2.2 Problema de otimização

A solução para o sistema (4.1.8) também pode ser encontrada resolvendo o problema de minimização

$$\begin{aligned} &\min_x \ \frac{1}{2}\|Ax - x\|^2 \\ &\text{s.t. } \langle x, e\rangle = 1. \end{aligned} \tag{4.2.2}$$

A restrição (4.2.2) garante a unicidade da solução. O problema pode ser reescrito na forma irrestrita e penalizada

$$\min_x \ f(x) \overset{def}{=} \frac{1}{2}\|Ax - x\|^2 + \frac{\gamma}{2}\left[\langle x, e\rangle - 1\right]^2, \tag{4.2.3}$$

onde $\gamma > 0$ é um parâmetro de penalidade associado à restrição (4.2.2). Se x^* é o vetor que satisfaz as Equações (4.1.1) e (4.1.8) então $f(x^*) = 0$, portanto x^* é minimizador de (4.2.3).

Observação sobre [19]

A formulação apresentada em [19] não leva em consideração o parâmetro μ. Isso impede a comparação com o método das potências e resulta em soluções ambíguas. O objetivo é classificar um conjunto de páginas web por ordem de importância. Se o problema (4.2.3) possui múltiplas soluções com diferentes ordenações para o conjunto de páginas o resultado obtido possui pouca utilidade.

4.2.3 Métodos de Busca em Coordenada

Note que o problema (4.2.3) é quadrático assim como o problema apresentado no Capítulo 3. A implementação para ambos os problemas é semelhante. Nesta Seção destacamos as diferenças considerando apenas blocos de tamanho unitário.

Constantes Lipschitz

A matriz Hessiana de f é

$$\nabla^2 f(x) = \nabla^2 f = (A - I)^T(A - I) + \gamma S$$

para todo $x \in \mathbb{R}^n$. Ao considerarmos blocos unitários a constante L_i corresponde ao i-ésimo elemento da diagonal de $\nabla^2 f$, assim

$$\begin{aligned} L_i &= \|A_i - e_i\|^2 + \gamma \\ &= \left\|(1-\mu)\bar{E}_i + \frac{\mu}{n}S_i - e_i\right\|^2 + \gamma \\ &= \left\|(1-\mu)\bar{E}_i + \frac{\mu}{n}e\right\|^2 - 2(1-\mu)\left\langle \bar{E}_i, e_i\right\rangle - 2\frac{\mu}{n}\langle e, e_i\rangle + \|e_i\|^2 + \gamma \\ &\overset{(4.1.6)}{=} \left\|(1-\mu)\bar{E}_i + \frac{\mu}{n}e\right\|^2 - 2\frac{\mu}{n} + 1 + \gamma \\ &= (1-\mu)^2\left\|\bar{E}_i\right\|^2 + 2(1-\mu)\frac{\mu}{n}\left\langle \bar{E}_i, e\right\rangle + \left(\frac{\mu}{n}\right)^2\|e\|^2 - 2\frac{\mu}{n} + 1 + \gamma \\ &\overset{(4.1.4)}{=} (1-\mu)^2\left\|\bar{E}_i\right\|^2 + 2(1-\mu)\frac{\mu}{n} + \frac{\mu^2}{n} - 2\frac{\mu}{n} + 1 + \gamma \\ &= (1-\mu)^2\left\|\bar{E}_i\right\|^2 + 2\frac{\mu}{n} - 2\frac{\mu^2}{n} + \frac{\mu^2}{n} - 2\frac{\mu}{n} + 1 + \gamma \\ &= (1-\mu)^2\left\|\bar{E}_i\right\|^2 - \frac{\mu^2}{n} + 1 + \gamma. \end{aligned}$$

Seja p_i a quantidade de elementos não nulos da coluna $\bar{E}_i$ para $i = 1, \ldots, n$, temos que

$$\left\|\bar{E}_i\right\|^2 = \sum_{j=1}^{n} (\bar{E}_{ji})^2 \overset{(4.1.3)}{=} p_i \left(\frac{1}{p_i}\right)^2 = \frac{1}{p_i}$$

e portanto

$$L_i = \frac{(1-\mu)^2}{p_i} - \frac{\mu^2}{n} + 1 + \gamma.$$

Logo, se os valores p_i estão prontamente disponíveis o calculo das constantes L_i não depende diretamente dos valores da matriz $\bar{E}$ sendo necessárias $O(1)$ operações para cada L_i, totalizando $O(n)$ operações.

Vetor residual

O gradiente de f é

$$\nabla f(x) = (A - I)^T (Ax - x) + \gamma \left[\langle e, x\rangle - 1\right] e.$$

A cada iteração é necessário o cálculo de uma componente de ∇f de maneira eficiente, isto é, aproveitando a estrutura esparsa do problema. Para o problema do Capítulo 3 esse cálculo é feito através da manutenção de um vetor residual que é atualizado, também de maneira eficiente, a cada iteração. No caso da matriz A existe uma componente densa S que impede a implementação direta da mesma estratégia utilizada no Capítulo 3. Porém a estrutura de S ainda pode ser aproveitada, notando que

$$Sx = \langle e, x\rangle e. \tag{4.2.4}$$

Considere o vetor residual $r(x) : \mathbb{R}^n \to \mathbb{R}^{n+2}$ dado por

$$r(x) = \begin{pmatrix} r_e(x) \\ r_s(x) \\ r_q(x) \end{pmatrix} = \begin{pmatrix} (1-\mu)\bar{E}x - x \\ \langle e, x\rangle \\ \langle e, r_e(x)\rangle \end{pmatrix}.$$

O vetor r é particionado em $(r_e, r_s, r_q) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}$ por conveniência. O gradiente ∇f pode ser reescrito em função r como (nas seguintes equações r_e, r_s e r_q denotam $r_e(x)$, $r_s(x)$ e $r_q(x)$ respectivamente)

$$\nabla f(x) = (A - I)^T \left(r_e + \frac{\mu}{n} r_s e\right) + \gamma \left(r_s - 1\right) e.$$

Seja $r'_s = \frac{\mu}{n} r_s$, a i-ésima componente de ∇f é

$$\begin{aligned}
\nabla_{x_i} f(x) &= \left\langle A_i - e_i,\, r_e + r'_s e\right\rangle + \gamma \left(r_s - 1\right) \\
&= \left\langle A_i,\, r_e + r'_s e\right\rangle - \left\langle e_i,\, r_e + r'_s e\right\rangle + \gamma \left(r_s - 1\right) \\
&= \left\langle A_i,\, r_e + r'_s e\right\rangle - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&= \left\langle (1-\mu)\bar{E}_i + \frac{\mu}{n} e,\, r_e + r'_s e\right\rangle - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&= (1-\mu)\left\langle \bar{E}_i,\, r_e + r'_s e\right\rangle + \frac{\mu}{n}\left\langle e, r_e + r'_s e\right\rangle - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&= (1-\mu)\left\langle \bar{E}_i,\, r_e + r'_s e\right\rangle + \frac{\mu}{n}\left(r_q + r'_s \|e\|^2\right) - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&= (1-\mu)\left(\left\langle \bar{E}_i,\, r_e\right\rangle + r'_s \left\langle \bar{E}_i, e\right\rangle\right) + \frac{\mu}{n}\left(r_q + r'_s n\right) - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&\overset{(4.1.4)}{=} (1-\mu)\left(\left\langle \bar{E}_i,\, r_e\right\rangle + r'_s\right) + \frac{\mu}{n} r_q + \mu r'_s - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&= (1-\mu)\left\langle \bar{E}_i,\, r_e\right\rangle + r'_s - \mu r'_s + \frac{\mu}{n} r_q + \mu r'_s - [r_e]_i - r'_s + \gamma \left(r_s - 1\right) \\
&= (1-\mu)\left\langle \bar{E}_i,\, r_e\right\rangle + \frac{\mu}{n} r_q - [r_e]_i + \gamma \left(r_s - 1\right)
\end{aligned}$$

O termo $\langle \bar{E}_i, r_e \rangle$ requer $O(p_i)$ operações para ser calculado e é o termo de maior custo computacional no calculo de $\nabla_{x_i} f$.

Resta mostrar que após a alteração de apenas uma componente de x podemos calcular de forma barata um novo vetor residual. Note que r é uma transformação linear, logo

$$r(x + te_i) = r(x) + tr(e_i).$$

O valor de cada componente de r em e_i é

$$\begin{aligned} r_e(e_i) &= (1-\mu)\bar{E}_i - e_i, \\ r_s(e_i) &= 1 \\ r_q(e_i) &= \langle e, (1-\mu)\bar{E}_i - e_i \rangle \\ &= (1-\mu)\langle e, \bar{E}_i \rangle - \langle e, e_i \rangle \\ &\overset{(4.1.4)}{=} 1 - \mu - 1 \\ &= -\mu. \end{aligned}$$

Assim

$$\begin{aligned} r_e(x + te_i) &= r_e(x) + t(1-\mu)\bar{E}_i - te_i, \\ r_s(x + te_i) &= r_s(x) + t \quad \text{e} \\ r_q(x + te_i) &= r_q(x) - t\mu. \end{aligned}$$

A nova componente r_e requer $O(p_i)$ operações para ser calculada.

Custo computacional

O Algoritmo CCG/UCG/BCG lista os passos para três variações do método de busca em coordenada aplicado ao problema (4.2.3); a busca em coordenada cíclica (CCG), busca em coordenada aleatória com probabilidade uniforme (UCG) e a busca em coordenada aleatória com probabilidade proporcional à constante Lipshitz (BCG). Os Passos com sufixo C, U e B são exclusivos ao métodos CCG, UCG e BCG respectivamente. A Tabela 4.1 lista a quantidade de operações exigidas em cada Passo do Algoritmo.

Passo	Operações
1	$O(nz)$
1B	$O(n)$
2C/2U/2B	$O(1)$
3	Em média $O(p)$
4	Em média $O(p)$
5	Amortizado $O(1)$

Tabela 4.1: Quantidade de operações exigidas em cada Passo do Algoritmo CCG/UDG/BCG.

Algoritmo CCG/UDG/BCG(Busca em Coordenada Cíclica/Uniforme/Enviesada (*Biased*)). Os dados de entrada são: a matriz $A \in \mathbb{R}^{n\times n}$ da forma (4.1.7); os parâmetros $\mu \in (0,1)$ e $\gamma > 0$, a precisão $\epsilon > 0$ e o iterando inicial x^0.

1. [Inicialização]. Inicialize os valores

$$\begin{aligned} L_i &\leftarrow \frac{(1-\mu)^2}{p_i} - \frac{\mu^2}{n} + 1 + \gamma, \quad i = 1, \ldots, n, \\ r^0 &\leftarrow r(x^0) \quad \text{e} \\ k &\leftarrow 0. \end{aligned}$$

1B. [Inicialização B]. Inicialize a tabela de distribuições substitutas.

2C. [Escolha Cíclica]. Faça

$$i_k = \begin{cases} 1 & \text{se } k = 0 \text{ ou } k = n, \\ i_{k-1} + 1 & \text{caso contrário.} \end{cases}$$

2U. [Escolha Uniforme]. Escolha i_k com probabilidade $Prob(i_k = i) = 1/n,\ i = 1, \ldots, n$.

2B. [Escolha Enviesada]. Escolha i_k com probabilidade

$$Prob(i_k = i) = \frac{L_i}{\sum_{j=1}^{n} L_j},\ i = 1, \ldots, n,$$

3. [Passo]. Faça

$$g_{i_k}^k \leftarrow (1-\mu)\left\langle \bar{E}_{i_k}, r_e^k \right\rangle + \frac{\mu}{n} r_q^k - \left[r_e^k\right]_{i_k} + \gamma\left(r_s^k - 1\right) \quad \text{e}$$

$$t_k \leftarrow -\frac{g_{i_k}^k}{L_{i_k}}.$$

4. [Atualização de Iterandos]. Faça

$$\begin{aligned} x^{k+1} &\leftarrow x^k + t_k e_{i_k}, \\ r_e^{k+1} &\leftarrow r_e^k + t_k\big((1-\mu)\bar{E}_{i_k} - e_{i_k}\big), \\ r_s^{k+1} &\leftarrow r_s^k + t_k \quad \text{e} \\ r_q^{k+1} &\leftarrow r_q^k - t_k\mu \end{aligned}$$

5. [Critério de parada]. Se k é múltiplo de n e

$$\frac{\left\|r_e^{k+1} - r_s^{k+1} e\right\|}{\|x^{k+1}\|} \leq \epsilon$$

pare e retorne x^{k+1}. Senão, faça $k \leftarrow k+1$ e volte para o Passo 2C, 2U ou 2B. ∎

4.2.4 Método de Máxima Descida com Armijo

Descrevemos agora o *método da máxima descida*, exposto na Seção 1, aplicado ao problema (4.2.3). A análise da taxa de convergência desse método pode ser encontrada em vários livros sobre otimização [3, 15, 21].

Algoritmo SAG(*Máxima Descida com Armijo*). Os dados de entrada são: a matriz $A \in \mathbb{R}^{n\times n}$ da forma (4.1.7); os parâmetro $\mu \in (0,1)$ e $\gamma > 0$, a precisão $\epsilon > 0$ e o iterando inicial x^0.

1. [Inicialização]. Faça $k \leftarrow 0$ e

$$r^0 \leftarrow (1-\mu)\bar{E}x^0 + \frac{\mu}{n}\left\langle e, x^0\right\rangle e - x^0.$$

2. [Residual]. Se

$$\frac{\left\|r^k\right\|_2}{\left\|x^k\right\|_2} \leq \epsilon$$

então pare e retorne x^k.

3. [Gradiente]. Faça

$$g^k \leftarrow (1-\mu)\bar{E}^T r^k + \frac{\mu}{n}\left\langle e, r^k\right\rangle e - r^k + \gamma\left[\left\langle e, x^k\right\rangle - 1\right] e \quad \text{e}$$
$$t \leftarrow 1.$$

4. [Busca Linear]. Faça

$$x' \leftarrow x^k - tg^k,$$
$$r' \leftarrow (1-\mu)\,\bar{E}x' + \frac{\mu}{n}\left\langle e, x'\right\rangle e - x' \quad \text{e}$$
$$f' \leftarrow \frac{1}{2}\left\|r'\right\|^2 + \frac{\gamma}{2}\left[\left\langle e, x'\right\rangle - 1\right]^2.$$

Se

$$f' \leq f_k - t\sigma\left\|g^k\right\|^2$$

faça

$$x^{k+1} \leftarrow x', \quad r^{k+1} \leftarrow r', \quad f_{k+1} \leftarrow f', \quad k \leftarrow k+1$$

e volte para o Passo 2. Caso contrário faça $t \leftarrow t/2$ e repita o Passo 4.

∎

4.3 Resultados Experimentais

4.3.1 Instâncias Aleatórias

Foram geradas instâncias aleatórias de dimensões $n \in \{262144,\ 524288,\ 1048576,\ 2097152\}$ e média de elementos não nulos por coluna iguais a $p \in \{10,\ 20\}$, no total de 8 instâncias. Os métodos implementados são listados na Tabela 4.2. Para os parâmetros μ e γ foram utilizados os valores

Método	Rótulo
Minimização cíclica (*Cyclic Minimization*)	CM
Descida aleatória uniforme (*Uniform Descent*)	UD
Descida aleatória enviesada (*Biased Descent*)	BD
Máxima descida com Armijo (*Steepest Armijo*)	SA
Método das potências (*Power Method*)	PM

Tabela 4.2: Métodos implementados.

$\mu \in \{0.01,\ 0.05,\ 0.15\}$ e $\gamma \in \{1/n,\ 1/\sqrt{n}\}$, totalizando 27 execuções por instância (o método das potências não é influenciado pelo parâmetro γ). Seguindo [19] o critério de parada utilizado em todas as execuções foi

$$\frac{\|Ax - x\|}{\|x\|} \leq \epsilon,$$

com $\epsilon = 0.01$. Os resultados são apresentados nas Tabelas 4.3 e 4.5.

Análise dos resultados

Em todos os casos o método PM atingiu o critério de parada em menor tempo. Teoricamente a performance do PM deveria ser pior para valores de μ próximos de zero, porém isso não foi observado em nenhuma das instâncias.

O método CM apresentou grande instabilidade, sua performance com $\gamma = n^{-1}$ é comparável ao PM, porém, com $\gamma = \sqrt{n^{-1}}$ o critério de parada não foi atingido dentro dos limites de iteração.

Os métodos BD e UD obtiveram performance semelhante, ambos os métodos atingiram o critério de parada em todos os casos.

O método SA obteve resultados piores em todos os casos. Os resultados omitidos correspondem à execuções em que o limite de iterações foi atingido (as iterações internas do método SA também foram limitadas).

		n	262144				524288			
		p	10		20		10		20	
			k	t(s)	k	t(s)	k	t(s)	k	t(s)
Método	γ	μ								
BD	n^{-1}	0.01	12	1.16	10	1.38	12	2.75	11	3.36
		0.05	12	1.17	10	1.39	12	2.72	11	3.32
		0.15	11	1.06	10	1.39	11	2.53	10	3.02
	$\sqrt{n^{-1}}$	0.01	21	2.02	16	2.19	20	4.54	16	5.23
		0.05	21	2.02	16	2.20	19	4.41	15	5.46
		0.15	19	1.84	15	2.05	18	4.06	15	4.48
CM	n^{-1}	0.01	4	0.21	3	0.30	4	0.48	3	0.59
		0.05	4	0.21	3	0.30	4	0.57	3	0.61
		0.15	4	0.24	3	0.30	4	0.58	3	0.60
	$\sqrt{n^{-1}}$	0.01	500	23.19	500	41.42	500	49.39	500	92.91
		0.05	500	23.21	500	41.68	500	52.81	500	90.91
		0.15	500	23.08	500	41.49	500	48.96	500	83.75
PM		0.01	4	0.15	3	0.21	4	0.29	3	0.41
		0.05	4	0.14	3	0.21	4	0.29	3	0.40
		0.15	4	0.15	3	0.21	4	0.29	3	0.41
SA	n^{-1}	0.01	385	22.37	313	32.70	385	45.27	313	63.55
		0.05	75	4.20	61	6.18	75	8.54	61	12.45
		0.15	23	1.32	20	2.04	24	2.73	20	4.07
	$\sqrt{n^{-1}}$	0.01	***	***	500	138.28	***	***	500	633.12
		0.05	113	14.96	93	22.76	500	335.71	500	561.81
		0.15	34	4.39	27	6.16	500	324.33	500	552.03
UD	n^{-1}	0.01	12	1.07	10	1.32	12	2.56	11	3.71
		0.05	12	1.08	10	1.31	12	2.56	11	3.91
		0.15	11	1.00	10	1.31	11	2.37	10	3.78
	$\sqrt{n^{-1}}$	0.01	20	1.78	15	1.97	21	4.41	17	4.84
		0.05	19	1.70	15	1.96	20	4.21	16	4.73
		0.15	18	1.61	14	1.83	19	3.98	16	5.05

Tabela 4.3: Resultado da execução dos métodos para instâncias de dimensão $n \in \{262144, 524288\}$.

		n	1048576				2097152			
		p	10		20		10		20	
			k	t(s)	k	t(s)	k	t(s)	k	t(s)
Método	γ	μ								
BD	n^{-1}	0.01	12	6.76	11	8.85	12	14.79	11	21.26
		0.05	12	6.75	10	8.08	12	14.87	10	19.45
		0.15	11	6.26	10	8.04	11	13.58	10	19.44
	$\sqrt{n^{-1}}$	0.01	22	12.71	17	13.46	22	26.65	18	33.88
		0.05	21	11.90	16	12.71	22	26.47	18	33.84
		0.15	20	11.17	16	12.62	20	24.16	17	32.08
CM	n^{-1}	0.01	4	1.35	3	1.56	4	3.45	3	5.19
		0.05	4	1.34	3	1.55	4	3.43	3	5.17
		0.15	4	1.31	3	1.54	4	3.43	3	5.17
	$\sqrt{n^{-1}}$	0.01	500	134.87	500	216.57	500	352.96	50	66.26
		0.05	500	133.51	500	216.12	500	353.42	50	66.64
		0.15	500	149.33	500	215.69	500	353.75	50	66.71
PM		0.01	4	1.01	3	1.21	4	3.21	3	5.07
		0.05	4	0.97	3	1.21	4	3.23	3	5.06
		0.15	4	0.99	3	1.21	4	3.21	3	5.05
SA	n^{-1}	0.01	385	139.33	313	197.94	50	59.96	50	115.00
		0.05	75	27.05	62	38.79	50	59.49	50	114.04
		0.15	24	9.10	20	12.83	24	29.09	20	46.56
	$\sqrt{n^{-1}}$	0.01	***	***	500	***	***	***	50	809.23
		0.05	122	112.31	95	151.08	50	384.22	50	750.96
		0.15	31	25.97	50	113.44	50	372.23	50	723.48
UD	n^{-1}	0.01	12	6.38	11	8.56	12	14.26	11	20.82
		0.05	12	6.37	10	7.80	12	14.22	11	20.75
		0.15	11	5.87	10	7.86	11	13.20	10	19.01
	$\sqrt{n^{-1}}$	0.01	21	11.22	17	12.96	21	24.36	16	29.58
		0.05	20	10.64	17	12.91	20	23.40	16	29.64
		0.15	18	9.50	16	12.26	19	22.50	15	27.85

Tabela 4.4: Resultado da execução dos métodos para instâncias de dimensão $n \in \{1048576,\ 2097152\}$.

4.3.2 Instâncias com dados reais

Os dados da Tabela 4.5 apresentam resultados da execução dos métodos sobre a instância com dados reais *web-google* [13]. Diferente dos resultados observados nas instâncias aleatórias o método das potências não foi o de melhor performance. Nesse caso a execução do método das potências falhou em duas das três execuções. O método CM falhou em duas das seis execuções. O método SA foi omitido da tabela por ter falhado em todas as execuções. Os métodos aleatórios UD e BD finalizaram com sucesso em todas as execuções.

		n	694656	
		p	6	
			k	t(s)
Método	γ	μ		
BD	n^{-1}	0.01	1961	519.07
		0.05	4420	1161.85
		0.15	2930	754.28
	$\sqrt{n^{-1}}$	0.01	1970	519.24
		0.05	4491	1157.07
		0.15	2934	768.99
CM	n^{-1}	0.01	982	89.41
		0.05	2213	210.55
		0.15	1442	136.69
	$\sqrt{n^{-1}}$	0.01	1414	130.16
		0.05	5000	***
		0.15	5000	***
PM		0.01	2047	125.46
		0.05	5000	***
		0.15	5000	***
UD		0.01	1964	476.90
		0.05	4432	1078.13
		0.15	2924	709.58
	$\sqrt{n^{-1}}$	0.01	1866	455.19
		0.05	4458	1088.09
		0.15	2910	709.68

Tabela 4.5: Resultado da execução dos métodos para instância com dados reais.

Capítulo 5

Máquina de Suporte Vetorial Linear

O problema da classificação de dados consiste em dado um conjunto de pares instância/rótulo $(x_i, y_i) \in \mathbb{X} \times \mathbb{Y},\ i = 1, \ldots, m$, sendo $\mathbb{Y}$ um conjunto discreto, encontrar uma relação $f : \mathbb{X} \to \mathbb{Y}$ que seja capaz de classificar corretamente instâncias ainda *desconhecidas*. Uma grande variedade de problemas pode ser reduzida ao caso $\mathbb{X} = \mathbb{R}^n$ e $\mathbb{Y} = \{+1, -1\}$. Nesse caso uma possível solução é encontrar um hiperplano de vetor normal $w \in \mathbb{R}^n$ e que satisfaça

$$\text{sign } \langle w, x_i \rangle = y_i,\ i = 1, \ldots, m. \tag{5.1}$$

Pode ser que tal hiperplano não exista, nessa caso dizemos que as instâncias x_i não são linearmente separáveis. Caso contrário, a quantidade hiperplanos separadores é possivelmente infinita. Uma técnica popular para a classificação de dados é a *máquina de suporte vetorial* (*Support Vector Machine*, SVM). Essa técnica consiste em encontrar o hiperplano que maximize distância entre as margens, definidas pelas instâncias mais próximas ao hiperplano separador. Essa distância é limitada por $1/\|w\|$, portanto o hiperplano pode ser encontrado resolvendo o problema

$$\begin{aligned} &\min_w \frac{1}{2}\|w\|^2 \\ &\text{s.a. } 1 - y_i \langle x_i, w \rangle < 0,\ i = 1, \ldots, m. \end{aligned} \tag{5.2}$$

O problema (5.2) pode ser reescrito na forma penalizada

$$\min_w f(w) = \frac{1}{2}\|w\|^2 + \gamma \sum_{i=1}^{m} \xi(w, x_i, y_i), \tag{5.3}$$

onde γ é um parâmetro de penalidade e ξ é uma função de erro que satisfaz $\xi(w, x_i, y_i) = 0$ se a i-ésima restrição em (5.2) é satisfeita pelo vetor w, caso contrário $\xi(w, x_i, y_i) > 0$. Uma função de erro possível é a *l2-loss*, definida como

$$\xi(w, x, y) = \frac{1}{2}\max\{1 - y \langle w, x \rangle,\ 0\}^2.$$

5.1 Implementações

Aplicamos variações do método de busca em coordenada ao problema

$$\min_w\ f(w) \stackrel{def}{=}\ \frac{1}{2}\|w\|^2\ +\ \frac{\gamma}{2}\sum_{i=1}^{m} \max\{\ 1 - y_i \langle x_i, w \rangle,\ 0\ \}^2. \tag{5.1.1}$$

Consideramos apenas blocos de tamanho unitário, isto é, uma componente do vetor de variáveis será alterada a cada iteração. Os iterandos são da forma

$$x^{k+1} = x^k - \frac{g^k_{i_k}}{L_{i_k}} e_{i_k} \tag{5.1.2}$$

onde i_k é o índice da variável que será alterada na k-ésima iteração, $g_i^k = \nabla_{x_i} f(x^k)$ e L_i é a constante Lipschitz associada à i-ésima coordenada.

Precisamos então determinar o valor das constantes L_i e determinar uma estratégia eficiente para o cálculo de uma componente do gradiente $\nabla f(x^k)$ a cada iteração.

Vetor residual

Para essas duas tarefas será conveniente reescrever a função f em termos do vetor residual $r : \mathbb{R}^n \to \mathbb{R}^m$ de valor

$$r(w) = e - YXw$$

onde

$$Y = \operatorname{diag}\{\, y_1, \cdots, y_m \,\} \quad \text{e} \quad X = \begin{bmatrix} x_1^T \\ \vdots \\ x_m^T \end{bmatrix} = \begin{bmatrix} X_1, & \cdots, X_n \end{bmatrix}.$$

A função f também pode ser escrita como

$$f(w) = \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\|r(w)_+\|^2, \tag{5.1.3}$$

onde $[x_+]_i = \max\{x_i, 0\},\ x \in \mathbb{R}^m$. Derivando a equação (5.1.1), em relação à w_i obtemos

$$\nabla_i f(w) = w_i - \gamma \sum_{j=1}^{m} y_j x_{ji} \max\{1 - y_i \langle x_i, w\rangle, 0\},$$

que também pode ser escrita com o vetor residual como

$$\nabla_i f(w) = w_i - \gamma \langle YX_i, r(w)_+ \rangle. \tag{5.1.4}$$

Da equação (5.1.4) fica claro que a operação de maior custo para o calculo de uma componente de ∇f é o produto interno $\langle YX_i, r(w)_+\rangle$. A estratégia para efetuar tal operação de maneira eficiente é semelhante àquela utilizada no Capítulo 3 para o problema de mínimos quadrados linear. Em vez do calculo completo do vetor $r^k = r(w^k)$ seu valor pode ser obtido de maneira eficiente a partir do vetor residual anterior pela expressão

$$r^{k+1} = r^k + \frac{g_{i_k}^k}{L_{i_k}} YX_{i_k}. \tag{5.1.5}$$

A quantidade de operações exigidas nas expressões (5.1.4) e (5.1.5) é proporcional à quantidade de elementos não-nulos do vetor X_i. Vamos denotar por p_i a quantidade de elementos não-nulos da coluna X_i, por $nz = \sum_{i=1}^n p_i$ a quantidade de elementos não-nulos da matriz X e por $p = nz/n$ a quantidade *média* de elementos não-nulos *por coluna* de X.

Constante Lipschitz

Resta agora determinar o valor das constantes $L_i,\ i = 1, \ldots, n$. Por conveniência lembramos aqui a seguinte definição: a função $f : \mathbb{R}^n \to \mathbb{R}$ é Lipshitz em relação ao i-ésimo valor de entrada se existe um valor $L_i > 0$ para o qual a inequação

$$\left|\nabla_i f(x + te_i) - \nabla_i f(x)\right| \le L_i |t|$$

é satisfeita para qualquer vetor x e escalar t. Ignorando o operador $[\cdot]_+$ na equação (5.1.3) temos uma função quadrática de matriz hessiana $I + \gamma X^T X$. Nesse caso o valor da constante seria

$$L_i = 1 + \gamma \|X_i\|^2, \tag{5.1.6}$$

que corresponde ao i-ésimo valor da diagonal da matriz hessiana. De fato, esse é o valor obtido ao aplicarmos a definição anterior à função f (eq. (5.1.3)):

$$\begin{aligned}\left|\nabla_i f(w+te_i) - \nabla_i f(w)\right| &= \left|t - \gamma \left\langle YX_i, r(w+te_i)_+ - r(w)_+\right\rangle\right| \\ &\leq |t| + \gamma\left|\left\langle YX_i, r(w+te_i)_+ - r(w)_+\right\rangle\right| \\ &\leq |t| + \gamma \left\|YX_i\right\| \left\| \left[r(w) - tYXe_i\right]_+ - r(w)_+ \right\| \\ &\leq |t| + \gamma \left\|YX_i\right\| \left\|tYX_i\right\| \\ &= \left(1 + \gamma \left\langle YX_i, YX_i\right\rangle\right)|t| \\ &\leq \left(1 + \gamma \left\|X_i\right\|^2\right)|t|\,.\end{aligned}$$

A igualdade é satisfeita quando $r(w+te_i)_+ = r(w+te_i)$, $r(w)_+ = r(w)$ e $t > 0$, portanto o valor da equação (5.1.6) é o menor possível.

5.1.1 Busca em Coordenada

O Algoritmo CDV/UDV/BDV detalha a implementação do método de busca em coordenada aleatória utilizando as informações descritas na Seção anterior.

Algoritmo CDV/UDV/BDV(Busca em Coordenada Cíclica/Uniforme/Enviesada (*Biased*)). Os dados de entrada são: a matriz $X \in \mathbb{R}^{m\times n}$; os rótulos $y_1, \ldots, y_m$; o parâmetro de penalidade $\gamma > 0$; o iterando inicial w^0 e a precisão $\epsilon > 0$.

1. [Inicialização]. Inicialize os valores

$$\begin{aligned}&L_i \leftarrow 1 + \gamma \left\|X_i\right\|^2,\ i = 1, \ldots, n, \\ &r^0 \leftarrow e - YXw^0 \quad \text{e} \\ &k \leftarrow 0.\end{aligned}$$

1B. [Inicialização B]. Inicialize a tabela de distribuições substitutas.

2C. [Escolha Cíclica]. Faça $i_k \leftarrow k \bmod n + 1$.

2U. [Escolha Uniforme]. Escolha i_k com probabilidade $Prob(i_k = i) = 1/n,\ i = 1, \ldots, n$.

2B. [Escolha Enviesada]. Escolha i_k com probabilidade

$$Prob(i_k = i) = \frac{L_i}{\sum_{j=1}^n L_j},\ i = 1, \ldots, n,$$

3. [Passo]. Faça

$$\begin{aligned}&g_{i_k}^k \leftarrow w_{i_k} - \gamma \left\langle YX_{i_k}, r_+^k \right\rangle \quad \text{e} \\ &t_k \leftarrow \frac{g_{i_k}^k}{L_{i_k}}.\end{aligned}$$

4. [Atualização de Iterandos]. Faça

$$\begin{aligned}&x^{k+1} \leftarrow x^k - t_k e_{i_k} \quad \text{e} \\ &r^{k+1} \leftarrow r^k + t_k YX_{i_k}.\end{aligned}$$

5. [Critério de parada]. Se $k \bmod n = 0$ calcule o gradiente ∎

5.1.2 Decréscimo Suficiente de Coordenada

Em [7] é apresentado um método baseado na alteração de apenas uma variável escolhida aleatoriamente a cada iteração. São duas diferenças principais em relação ao método da seção anterior: (1) é empregada uma versão generalizada da hessiana de f para o cálculo do passo e (2) a escolha da coordenada é baseada em permutação. Essas duas diferenças e suas consequências são apresentadas a seguir.

Calculo do iterando

Os iterandos são da forma

$$x^{k+1} = x^k - t_k \frac{g^k_{i_k}}{h^k_{i_k}} e_{i_k},$$

onde e $h^k_i = \nabla^2_{ii} f(w^k)$. A função f não é duas vezes diferenciável, a seguinte definição para $\nabla^2_{ii} f$ é utilizada:

$$\nabla^2_{ii} f(w) = 1 + \gamma \sum_{j \in I(w)} x^2_{ji}$$

onde

$$I(w) = \{\ i\ :\ i = 1, \ldots, m,\ r_i(w) > 0\ \}$$

(note que apenas os valores correspondentes à diagonal da matriz hessiana são definidos).

O passo dado na equação (5.1.2) é potencialmente maior se comparado ao passo (5.1.2), pois $\nabla^2_{ii} f(w) \leq L_i$, o que pode levar à redução mais rápida do valor da função objetivo. Mas esse passo maior também significa que a garantia teórica de convergência apresentada por Nesterov não se aplica ao método cdper. A garantia da convergência do método cdper é baseada no decréscimo da função objetivo

$$f(x) - f(x - z e_i) \geq -\sigma z^2.$$

Algoritmo CDPER(*Minimização Inexata de Coordenada Permutada*). Os dados de entrada são: a matriz $X \in \mathbb{R}^{m \times n}$; os rótulos $y_1, \ldots, y_m$; o parâmetro de penalidade $\gamma > 0$; o iterando inicial w^0 e a precisão $\epsilon > 0$.

1. [Inicialização]. Inicialize os valores

$$\begin{aligned} &L_i \leftarrow 1 + \gamma \left\| X_i \right\|^2,\ i = 1, \ldots, n, \\ &\pi_i \leftarrow i,\ i = 1, \ldots, n, \\ &r^0 \leftarrow e - YXw^0 \quad \text{e} \\ &k \leftarrow 0. \end{aligned}$$

2. [Permutação]. Permute os elementos π_i e π_j onde

$$i = k \bmod n + 1, \quad j = \text{rand}\ \{i,\ i+1,\ \ldots,\ n\}$$

e rand S denota a escolha aleatória uniforme de um elemento do conjunto finito S. Faça

$$i_k \leftarrow \pi_i.$$

3. [Passo]. Calcule

$$d^k \leftarrow -\frac{g_{i_k}}{h_{i_k}}$$

onde

$$g_i = w^k_i - \gamma \left\langle Y X_i,\, r^k_+ \right\rangle, \quad h_i = 1 + \gamma \sum_{j\, \in\, I^k} x^2_{ji} \quad \text{e} \quad I^k = \{\ i\ :\ i = 1, \ldots, n,\ r^k_i > 0\ \}.$$

4a. [Testa Necessidade de Busca Linear]. Faça $t_k \leftarrow 1$. Se

$$t_k \leq \frac{h_{i_k}}{L_{i_k}/2 + \sigma}$$

vá para o Passo 5.

4b. [Busca Linear]. Faça

$$t_k \leftarrow \min A \cap B \tag{5.1.7}$$

onde

$$A = \{1,\ 2^{-1},\ 2^{-2}\} \quad \text{e}$$
$$B = \{\ t\ :\ f^k - f(w^k + td_k e_{i_k}) \leq -\sigma t^2\ \}$$

5. [Atualização de iterandos]. Faça

$$x^{k+1} \leftarrow x^k + t_k d_k e_{i_k} \quad \text{e}$$
$$r^{k+1} \leftarrow r^k + t_k d_k Y X_{i_k}$$

6. [Critério de parada]. Se k é múltiplo de n e calcule

5.2 Resultados Experimentais

5.2.1 Instâncias

Para comparação das implementações foram utilizadas três instâncias para o problema de classificação de texto:

news20 [11]: Uma coleção de 20000 mensagens extraídas de 20 *newsgroups*. A versão original do conjunto de dados possui 20 rótulos, um para cada newsgroup. A versão utilizada neste trabalho contém apenas dois rótulos, sendo cada um correspondente à 10 dos rótulos originais. Cada instância consiste em um vetor de frequência binária normalizado.

rcv1 [14]: Composta por uma coleção de artigos da agência de notícias *Reuters*.

real-sim [17]: Coleção mais de 73,000 artigos também com rótulos binários.

Essas instâncias, que podem ser obtidas em `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/` , são as mesmas utilizadas em [7] para comparação do algoritmo CDPER com outros métodos. Algumas das instâncias utilizadas em [7] não estão publicamente disponíveis e por isso não foram utilizadas neste trabalho.

inst.	m	n	nz/n	maior L_i	menor L_i
news20	19996	1355191	6	625	1
real-sim	72309	20958	176	1104	1
rcv1	697641	47236	1080	23725	1

Tabela 5.1: Informações sobre as instâncias. As colunas nz/n, maior L_i e menor L_i apresentam valores aproximados. A penalidade $\gamma = 8$ foi utilizada para cálculo das constantes L_i.

A Tabela 5.1 contém informações sobre as dimensões sobre instâncias utilizadas. A informação sobre a maior e menor constante L_i é relevante para a execução do algoritmo *bd*. Esse valores indicam que existem coordenadas que serão selecionadas com maior frequência que outras. No caso da instância, por exemplo, news20 a coordenada mais selecionada será alterada seiscentas vezes mais que a menos selecionada.

5.2.2 Experimentos

Para cada implementação e cada instância foram realizados três experimentos. Os resultados desses experimentos podem ser observados nas Figuras 5.1, 5.2 e 5.3.

A Figura 5.1 mostra a taxa de acerto da SVM sobre o conjunto de teste, esse é o resultado de maior interesse prático. A escolha do parâmetro de penalidade γ é um passo importante no treino de uma SVM, uma escolha inapropriada pode fazer com que a taxa de acerto seja insatisfatória. Portanto, para comparação da taxa de acerto cada método foi executado com o valor da penalidade γ mais apropriado (Tabela 5.2).

A escolha de penalidade foi feita utilizando a estratégia de validação cruzada *5-fold*. Essa estratégia consiste em, para cada valor candidato de penalidade, realizar os seguintes passos: particionar o conjunto de treino em 5 partes; selecionar uma das partes para ser utilizada como conjunto de teste; utilizar as demais 4 partes para treino da SVM; utilizar o conjunto de teste para validar a SVM obtida; repetir esse processo utilizando cada uma das 5 partes como conjunto de teste. Importante destacar que o conjunto de teste não é utilizado nessa etapa.

método	bd		cd		sd		ud	
	γ	Prec.	γ	Prec.	γ	Prec.	γ	Prec.
inst.								
news20	10	97,20	10	97,15	7	97,23	12	97,18
rcv1	10	97,64	10	97,62	4	97,73	5	97,72
real-sim	10	97,10	10	97,32	7	97,10	18	97,03

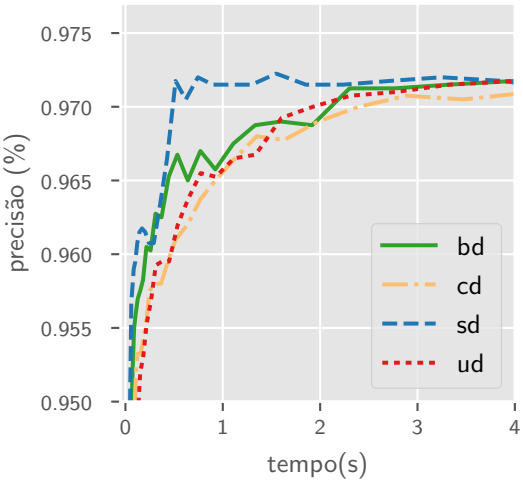
Tabela 5.2: Penalidade utilizada e precisão alcançada por cada um dos métodos em cada uma das instâncias.

Podemos observar que para todos os método e em todas as instâncias a diferença de precisão obtida é inferior à 1%. Em todas as instâncias o algoritmo sd chega mais rápido ao resultado à precisão final.

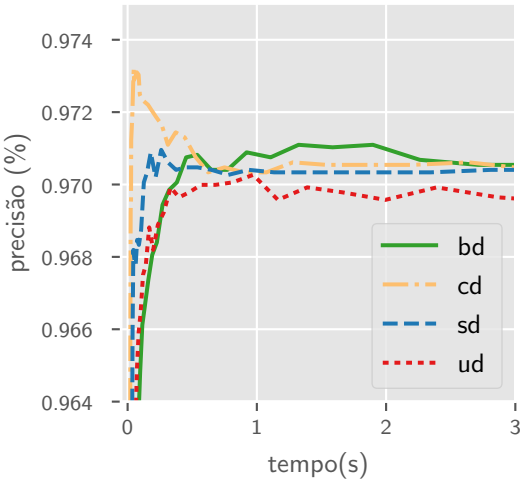
No outro experimento estamos interessados em comparar a capacidade dos métodos de minimizar a função objetivo. Para esse experimento utilizamos o valor $\gamma = 8$, já que estamos interessados no valor da função objetivo e não na capacidade de predição. A Figura 5.2 mostra a redução da valor objetivo em relação ao valor inicial $w^0 = 0$. Observamos que todos os métodos possuem capacidade similar de minimizar a função objetivo, sendo que o método sd o mais rápido.

Nesse mesmo experimento também acompanhamos evolução da norma do gradiente, Figura 5.3. Nesse caso a superioridade do método sd é evidente.

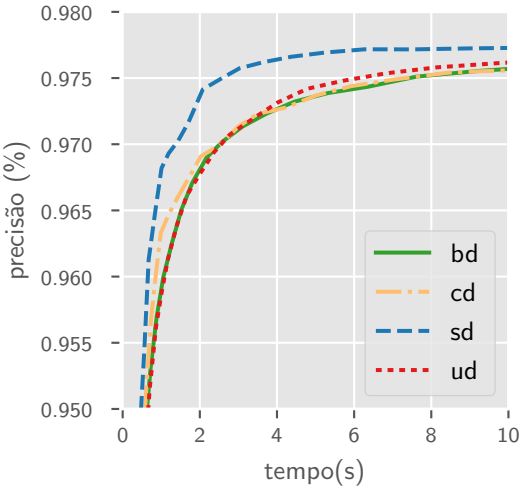
A partir destes experimentos podemos observar que o método sd é superior em minimizar a norma do gradiente. Mas essa superioridade não é refletida de maneira considerável na minimização do valor da função objetivo e, principalmente, na capacidade de predição do modelo, que é o critério mais importante.



(a) news20

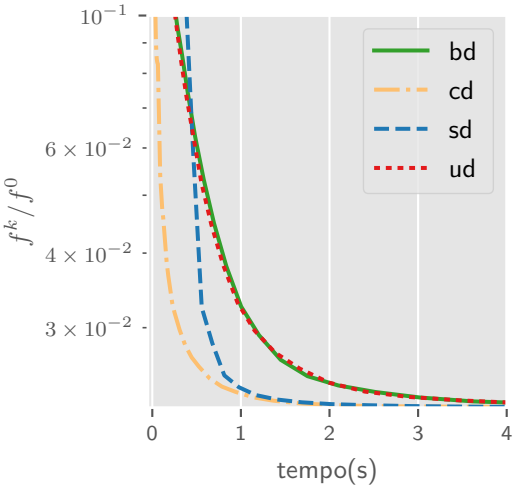


(b) real-sim

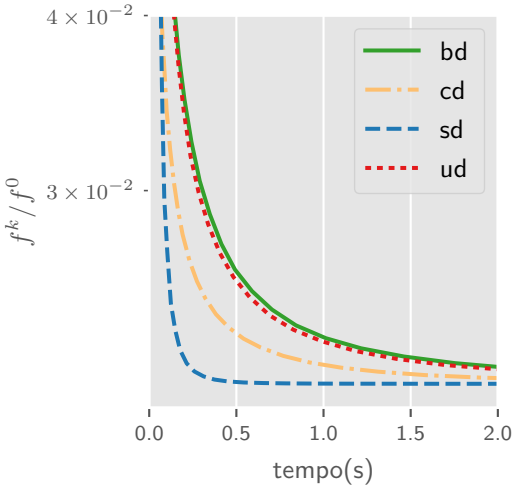


(c) rcv1

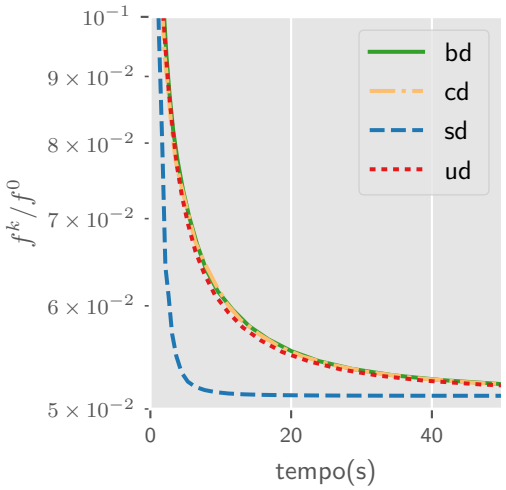
Figura 5.1: Taxa de acerto sobre o conjunto de testes para cada uma das instâncias.



(a) news20

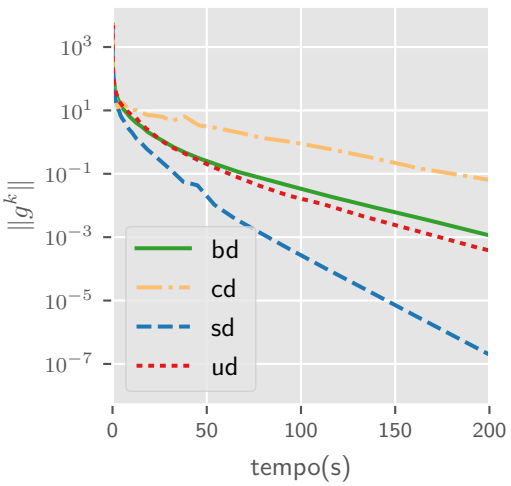


(b) real-sim

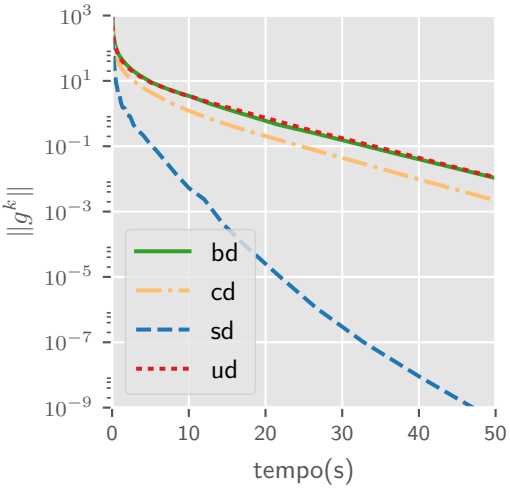


(c) rcv1

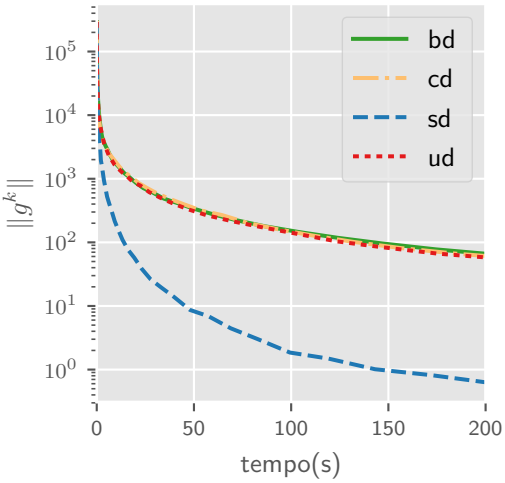
Figura 5.2: Valor da função objetivo em relação ao valor inicial em função do tempo.



(a) news20



(b) real-sim



(c) rcv1

Figura 5.3: Norma do gradiente em função do tempo.

Capítulo 6

Considerações finais e trabalhos futuros

Neste trabalho estudamos métodos iterativos de otimização de funções convexas caracterizados pela alteração de um número reduzido de variáveis a cada iteração. O foco foi dado ao método descrito em [19] e na sua aplicação e comparação com (a) outros métodos baseados na alteração de poucas coordenadas a cada iteração e (b) métodos baseados no cálculo do gradiente completo a cada iteração. Os problemas atacados neste trabalho possuem duas características principais: esparsidade dos dados envolvidos e grande número de variáveis.

Um dos problemas atacados foi o de quadrados mínimos lineares. Neste caso os experimentos mostraram que métodos de busca em coordenada possuem capacidade semelhante de minimização da função objetivo quando comparados ao método máxima descida utilizando *backtracking* e a condição de Armijo. Quando comparado com um método gradiente mais elaborado, neste caso o método do gradiente espectral [5], os métodos de busca em coordenada apresentaram vantagem inicial na minimização da função objetivo mas se mostraram inferiores quanto maior o tempo de execução. Também comparamos métodos de busca em coordenada variando a quantidade de variáveis alteradas a cada iteração. Nesse sentido observamos que quando a alteração consiste na minimização exata do sub-problema formado apenas pelas variáveis selecionadas nenhuma vantagem é obtida. A minimização exata é a forma como o método de busca em coordenada é descrito com maior frequência na literatura. Trabalhos recentes descrevem métodos em que a alteração das variáveis é feita de acordo com a derivada parcial da função objetivo. Nesse caso, o aumento da quantidade de variáveis alteradas a cada iteração mostrou ter efeito positivo, porém não tão significativo, na resolução do problema.

Outro problema atacado, conhecido como *google-problem*, consiste em atribuir importância à páginas *web*. Esse é o mesmo problema atacado em [19], porém neste trabalho a formulação do problema é mais próxima à aplicada na prática. Essa formulação é aumentou consideravelmente a dificuldade da aplicação do método de busca em coordenada ao problema, porém nos permitiu comparação com o *pagerank*, o algoritmo baseado no método das potências para ordenação de páginas. Nessa comparação observamos que, tanto para instâncias aleatórias quanto uma instância com dados reais, métodos de busca em coordenada apresentaram resultados competitivos em relação ao *pagerank*.

Finalmente, comparamos o método descrito em [19] com outro método baseado na alteração de coordenadas unitárias aplicado à classificação de documentos. Nesse caso o método descrito em [19] é de mais simples implementação e apresentou capacidade de classificação dos documentos equiparável à do método de mais difícil implementação e elaborado especificamente para o problema.

Apêndice A

Método das distribuições substitutas

A cada iteração do algoritmo descrito por Nesterov [19] é necessário o sorteio de um elemento de um conjunto finito. Em uma versão do algoritmo esse sorteio é feito de forma uniforme, isto é, os elementos são sorteados com a mesma probabilidade. Nesse caso o sorteio pode ser feito com uma quantidade constante de operações (como será descrito mais adiante). Em outra versão do algoritmo cada elemento possui uma probabilidade arbitrária de ser sorteado. Nesterov sugere uma estratégia para a realização do sorteio que requer $O(\log n)$ operações. Em muitas situações esse custo é aceitável, no entanto é possível realizar o mesmo sorteio com $O(1)$ operações utilizando o *método das distribuições substitutas*.

Seja U um valor aleatório escolhido uniformemente no intervalo $[0,\, 1)$, isto é, U satisfaz a propriedade

$$\text{Prob}(\, a < U < b\,) = b - a, \quad a,\, b \in [0,\, 1). \tag{A.0.1}$$

O sorteio uniforme de um valor inteiro entre 1 e n pode obtido pela expressão

$$X = \lfloor nU \rfloor + 1.$$

A variável X é um valor inteiro entre 1 e n satisfazendo $\text{Prob}(\, X = i\,) = 1/n,\ i = 1, \ldots, n$. A expressão anterior pode ser reescrita da seguinte maneira:

$$X = \begin{cases} 1, & \text{se } 0 \leq U < 1/n; \\ 2, & \text{se } 1/n \leq U < 2/n; \\ \vdots & \\ n, & \text{se } (n-1)/n \leq U < 1. \end{cases}$$

Nessa forma, podemos generalizar a expressão para considerar probabilidades arbitrárias para a escolha de cada elemento. Seja p_i a probabilidade de que $X = i$ para $i = 1, \ldots, n$, X pode ser definido a partir do valor U por

$$X = \begin{cases} 1, & \text{se } 0 \leq U < p_1; \\ 2, & \text{se } p_1 \leq U < p_1 + p_2; \\ 3, & \text{se } p_1 + p_2 \leq U < p_1 + p_2 + p_3; \\ \vdots & \\ n, & \text{se } p_1 + p_2 + \cdots + p_{n-1} \leq U < 1. \end{cases}$$

Seja $P_1 = 0$ e $P_i = P_{i-1} + p_{i-1}$ para $i = 2, \ldots, n+1$, obviamente $P_{n+1} = 1$.

O método das distribuições substitutas (*Alias Method* [12, 26]) consiste em substituir a distribuição arbitrária p de n elementos por uma distribuição uniforme de n elementos e n distribuições arbitrárias de 2 elementos.

Nesse método são necessários dois valores aleatórios uniformes U_1 e U_2. A escolha do valor X é feita da seguinte maneira

$$Y = \lfloor U_1 n \rfloor + 1,$$
$$X = \begin{cases} Y\,, & \text{se } q_i < U_2; \\ A_Y\,, & \text{caso contrário.} \end{cases}$$

onde $A \in \{1, \ldots, n\}^n$ é a tabela de substitutos e q_i representa de maneira implícita uma n distribuições de 2 elementos. O valor q_i é a probabilidade de escolha de Y e $1 - q_i$ é a probabilidade de escolha do valor A_Y.

Os valores A e q devem satisfazem a propriedade

$$p_i = \frac{1}{n}\left(q_i + \sum_{j=1,\ A_j=i}^{n} (1 - q_i)\right).$$

Assim, a principal etapa do método das distribuições substitutas é a construção da tabela A e do vetor q. O algoritmo *ALIAS* a seguir descreve uma maneira de construir tais valores com $O(n)$ operações.

Algoritmo ALIAS(*Construção da tabela de distribuições substitutas*). A entrada é a distribuição de probabilidades p de n elementos. A saída é a tabela A e o vetor $q \in \mathbb{R}$.

1. [Inicialização]. Faça

$$A \leftarrow (1,\ 2,\ \ldots,\ n),$$
$$q \leftarrow np,$$
$$S \leftarrow \{i : q_i < 1,\ i = 1, \ldots, n\} \text{ e}$$
$$G \leftarrow \{i : i \notin S, i = 1, \ldots, n\}.$$

2. [Defina o substituto]. Se $|S| > 0$ selecione arbitrariamente elementos $g \in G$, $s \in S$ e faça

$$A_s \leftarrow g \text{ e}$$
$$q_g \leftarrow q_g - (1 - q_s).$$

3. [Atualize S e G]. Faça

$$S \leftarrow S \setminus \{s\}.$$

Se $q_g < 1$ faça

$$G \leftarrow G \setminus \{g\} \quad \text{e} \quad S \leftarrow S \cup \{g\}.$$

4. [Parada] Se $|S| = 0$ retorne A e q como resultado. Caso contrário volte ao passo 2.

∎

Referências Bibliográficas

[1] Auslender, A. *Optimisation: méthodes numériques*. Masson, 1976.

[2] Beck, A., and Tetruashvili, L. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization 23*, 4 (2013), 2037–2060.

[3] Bertsekas, D. *Nonlinear Programming*. Athena Scientific, 1999.

[4] Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[5] Birgin, E. G., Martínez, J. M., Raydan, M., et al. Spectral projected gradient methods: review and perspectives. *J. Stat. Softw 60*, 3 (2014), 539–559.

[6] Bryan, K., and Leise, T. The $25,000,000,000 eigenvector: The linear algebra behind google. *SIAM Review 48*, 3 (2006), 569–581.

[7] Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research 9*, Jul (2008), 1369–1398.

[8] d'Esopo, D. A convex programming procedure. *Naval Research Logistics Quarterly 6*, 1 (1959), 33–42.

[9] Friedman, J., Hastie, T., Höfling, H., Tibshirani, R., et al. Pathwise coordinate optimization. *The Annals of Applied Statistics 1*, 2 (2007), 302–332.

[10] HSL. A collection of fortran codes for large scale scientific computation. `http://www.hsl.rl.ac.uk`.

[11] Keerthi, S. S., and DeCoste, D. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research 6*, Mar (2005), 341–361.

[12] Knuth, D. E. The art of computer programming, 3rd edn. seminumerical algorithms, vol. 2, 1997.

[13] Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics 6*, 1 (2009), 29–123.

[14] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research 5*, Apr (2004), 361–397.

[15] Luenberger, D. G. *Linear and Non-Linear Programming*. 1989.

[16] Luo, Z.-Q., and Tseng, P. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications 72*, 1 (1992), 7–35.

[17] McCallum, A. Simulated/real/aviation/auto usenet data. `https://people.cs.umass.edu/~mccallum/data.html`. Accessed: 2010-09-30.

[18] Motwani, R., and Raghavan, P. *Randomized Algorithms*. Cambridge University Press, 1997.

[19] Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization 22*, 2 (2012), 341–362.

[20] Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, vol. 87. Springer Science & Business Media, 2013.

[21] Nocedal, J., and Wright, S. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[22] Ortega, J. M., and Rheinboldt, W. C. *Iterative solution of nonlinear equations in several variables*, vol. 30. Siam, 1970.

[23] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: bringing order to the web.

[24] Richtárik, P., and Takác, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program. 144*, 1-2 (2014), 1–38.

[25] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.

[26] Walker, A. J. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters 10*, 8 (1974), 127–128.

[27] Warga, J. Minimizing certain convex functions. *Journal of the Society for Industrial & Applied Mathematics 11*, 3 (1963), 588–593.

[28] Wright, S. J. Coordinate descent algorithms. *Mathematical Programming 151*, 1 (2015), 3–34.