Pontifícia Universidade Católica
do Rio de Janeiro

Luisa Silveira Rosa

Otimização Inversa via Online Learning

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Marco Serpa Molinaro

Rio de Janeiro
Junho de 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Luisa Silveira Rosa

Otimização Inversa via Online Learning

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marco Serpa Molinaro
Orientador
Departamento de Informática – PUC-Rio

Prof. Eduardo Sany Laber
Departamento de Informática – PUC-Rio

Prof. Ruy Luiz Milidiú
Departamento de Informática – PUC-Rio

Rio de Janeiro, 19 Junho 2019

Luisa Silveira Rosa

Bacharel em Sistemas de Informação (2017) pela Universidade Federal do Acre (UFAC).

Ficha Catalográfica

Luisa Silveira Rosa

Otimização Inversa via Online Learning / Luisa Silveira Rosa; orientador: Marco Serpa Molinaro. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2019.

v., 52 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Aprender a função objetivo. 2. Aprender as restrições. 3. Otimização Inversa. 4. Online Learning. 5. Follow the Regularized Leader.
I. Marco Serpa Molinaro. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Primeiramente a Deus, o autor de tudo, pela vida, pelos familiares e amigos, e, que com sua misericórdia e amor permitiu a realização deste trabalho. Aos meus pais que sempre me apoiaram e me incentivaram mesmo nos momentos mais difíceis. Aos meus irmãos que na minha ausência, provinda dos estudos, foram compreensivos e me apoiaram, enfim a toda minha família pelo apoio e cuidado. Ao meu orientador professor Marco Serpa Molinaro pela orientação, profissionalismo e valiosas discussões durante todo esse trabalho. A Pontifícia Universidade Católica do Rio de Janeiro, ao Departamento de Informática e aos professores que participaram da minha formação. Aos amigos que estiveram ao meu lado no decorrer desses dois anos em especial André, Rômulo, Micaele, Dalai, Lauro, Vinicius, Raul, Alysson, Matheus, Georges, Guilherme e Luiz. Enfim, a todos que contribuíram direta ou indiretamente para a realização deste trabalho agradeço ao Criador e Senhor Deus pela oportunidade de tê-los em minha vida.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

Resumo

Luisa Silveira Rosa; Marco Serpa Molinaro. **Otimização Inversa via Online Learning**. Rio de Janeiro, 2019. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Demonstramos como aprender a função objetivo e as restrições de problemas de otimização enquanto observamos sua solução ótima no decorrer de múltiplas rodadas. Nossa abordagem é baseada em técnicas de Online Learning e funciona para funções objetivo lineares sob conjuntos viáveis arbitrários generalizando trabalhos anteriores. Os dois algoritmos, um para aprender a função objetivo e o outro par aprender as restrições, convergem a uma taxa de $O(\frac{1}{\sqrt{T}})$ que nos permitem produzir soluções tão boas quanto as ótimas em poucas observações. Finalmente, mostramos a eficácia e possíveis aplicações de nossos métodos em um amplo estudo computacional.

Palavras-chave

Aprender a função objetivo; Aprender as restrições; Otimização Inversa; Online Learning; Follow the Regularized Leader.

Abstract

Luisa Silveira Rosa; Marco Serpa Molinaro (Advisor). **Inverse Optimization via Online Learning**. Rio de Janeiro, 2019. 52p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

We demonstrate how to learn the objective function and constraints of optimization problems while observing its optimal solution over multiple rounds. Our approach is based on Online Learning techniques and works for linear objective functions under arbitrary feasible sets by generalizing previous work. The two algorithms, one to learn objective function and other to learn constraints, converge at a rate of $O(\frac{1}{\sqrt{T}})$ that allow us to produce solutions as good as the optimal in a few observations. Finally, we show the efficacy and possible applications of our methods in a significant computational study.

Keywords

Learning Objective Functions; Learning Constraints; Inverse Optimization; Online Learning; Follow the Regularized Leader.

Sumário

1 Introdução **10**
1.1 Otimização Inversa Online 11
1.2 Nossos Resultados 13

2 Literatura em Otimização Inversa **15**
2.1 Otimização Inversa Offline em geral 15
2.2 Otimização Inversa Linear 16
2.3 Otimização Combinatória Inversa 16
2.3.1 Problema do caminho mais curto inverso 17
2.3.2 Problema da Árvore geradora mínima inversa 17
2.3.3 Problema da mochila inverso 18
2.4 Otimização Inversa Cônica/Convexa 19
2.5 Otimização Inversa Inteira 20
2.6 Otimização Inversa Online 21

3 Preliminares: *Online Learning* **25**
3.1 *Multiplicative Weights Update* 26
3.2 *Follow the Regularized Leader* 27

4 Aprender a Função Objetivo **29**
4.1 Algoritmo inicial: assumindo diâmetros conhecidos 30
4.2 Algoritmo AlgoFO 31

5 Aprender as Restrições **34**
5.1 Prova do Teorema 10 35

6 Experimentos computacionais **38**
6.1 Aprender Preferências dos Clientes 38
6.2 Aprender Tempos de Viagem 40
6.3 Aprender Custos de Transporte 42
6.4 Aprender Custos de Produção 44
6.5 Aprender Capacidades de Recurso 46

7 Conclusões **48**

Referências bibliográficas **49**

Lista de figuras

Figura 1.1 Visão geral do problema OInvOnline-FO 11
Figura 1.2 Visão geral do problema OInvOnline-LD 13

Figura 4.1 Iteração do AlgoDiamFO com o FTRL 31

Figura 6.1 Em (a) plotamos o erro de solução em cada tempo t e o erro de solução médio do algoritmo AlgoFO, assim como em (b) para o algoritmo OOFL 39
Figura 6.2 Plotamos o gap multiplicativo entre as soluções $\bar{x}_t$ e x_t na função objetivo c_{true}, dada por $\frac{\sum_{t'=1}^{t} c_{true}\bar{x}_{t'}}{\sum_{t'=1}^{t} c_{true}x_{t'}}$ em (a) para o AlgoFO, assim como em (b) para o OOFL 40
Figura 6.3 Erro e o erro médio em cada instante t para o problema de aprender os tempo de viagem. 42
Figura 6.4 Erro de solução em cada instante $tv \in T$ e o erro médio para os algoritmos AlgoFO, em (a) e OOFL, em (b). 44
Figura 6.5 Gap multiplicativo entre as soluções $\bar{x}_t$ e x_t na função objetivo c_{true} para o AlgoFO (a) e para o OOFL (b). 44
Figura 6.6 Erro por iteração t e o erro médio para o algoritmo AlgoFO em (a) e para o algoritmo OOFL-OGD em (b). Em (c) plotamos o gap multiplicativo entre as soluções $\bar{x}_t$ e x_t para o algoritmo AlgoFO e em (d) para o algoritmo OOFL-OGD. 46
Figura 6.7 Em (a) plotamos o erro em cada tempo t e o erro médio do algoritmo , assim como em (b) o gap multiplicativo entre as soluções $\bar{x}_t$ e x_t, dada por $\frac{\sum_{t'=1}^{t} c_{t'}\bar{x}_{t'}}{\sum_{t'=1}^{t} c_{t'}x_{t'}}$. 47

Mas Deus prova o seu amor para conosco, em que Cristo morreu por nós, sendo nós ainda pecadores.

Romanos 5:8.

1
Introdução

O problema de otimização inversa pode ser visto como o complemento de um problema de otimização tradicional. Na otimização tradicional deseja-se encontrar a solução x^* em um conjunto viável X que maximize $f(c, x)$, onde c parametriza a função objetivo:

$$\begin{aligned} \max \quad & f(c, x) \\ \text{s.t.} \quad & x \in X. \end{aligned} \tag{1-1}$$

Tome como exemplo o problema de encontrar o caminho mais curto entre os nós s e t de um grafo $G = (V, E)$. Neste caso X modela os possíveis caminhos $s - t$ do grafo e c é o vetor de custos das suas arestas.

Por outro lado, a otimização inversa consiste em determinar valores para parâmetros dos problemas de otimização, como o parâmetro c e a região viável X, dada uma solução ótima conhecida $\hat{x} \in X$. Por exemplo, dado um caminho de menor custo $\hat{x}$ entre os nós s e t de um grafo G, a otimização inversa busca inferir os custos c atribuídos as arestas do grafo.

Mais formalmente, o problema de otimização inversa, na versão estática, para inferir a função objetivo (em uma família paramétrica) é definido da seguinte forma. Dado o conjunto de soluções viáveis X, uma solução $\hat{x}$, opcionalmente uma estimativa $\hat{c}$ do parâmetro c, e uma função de perda ϕ, deseja-se encontrar um parâmetro c tal que $\hat{x}$ seja ótima com relação a esse parâmetro e a perda $\phi(c, \hat{c})$ seja mínima:

$$\begin{aligned} \min \quad & \phi(c, \hat{c}) \\ \text{s.t.} \quad & f(c, \hat{x}) = \max_{x \in X} f(c, x). \end{aligned} \tag{1-2}$$

Esse problema modela, por exemplo, a tarefa de aprender a função utilidade de consumidores através de observações de seus comportamentos (sob a hipótese de que os consumidores maximizam suas utilidades).

Otimização inversa é aplicada em áreas como finanças (1), transporte (2, 3), saúde (4, 5), meio ambiente, setor elétrico (6), planejamento de produção (4, 5, 7), leilões (8), tomografia sísmica e computadorizada (9–12). Diversos métodos são disponíveis para resolver diferentes problemas de otimização inversa, como problemas de programação linear (4, 5, 7, 13–17), otimização

combinatória (9–12, 18–23), convexa (24–26), cônica (1), inteira (27–30), *Markov Decision Processes* (MDP) (31) e otimização online (32–34). No Capitulo 2 vamos expor com mais detalhes os métodos disponíveis para se resolver problemas de otimização inversa bem como algumas de suas aplicações.

1.1 Otimização Inversa Online

A versão estática do problema de otimização inversa apresenta suas limitações. No exemplo de aprendizado da função objetivo de consumidores mencionado acima, tipicamente as informações são obtidas ao longo do tempo, e um componente importante é o quão rápido o aprendizado é realizado.

Para modelar essa situação, recentemente foi introduzida a versão online do problema de otimização inversa (32); esse é o problema central considerado no nosso trabalho. Para simplificar, apresentamos separadamente duas versões desse problema: na primeira o objetivo é aprender a função objetivo (OInvOnline-FO) e no segundo aprender a região viável (OInvOnline-LD).

OInvOnline-FO Esse problema pode ser descrito da seguinte forma: Existe um vetor c_{true} **desconhecido** que parametriza a função objetivo. No tempo t, o algoritmo observa a região viável desse tempo X_t. Nesse momento, ele tem que produzir uma estimativa c_t da função objetivo c_{true}. Depois disso, o algoritmo obtém uma solução $x_t \in X_t$ ótima com relação a c_{true}:

$$\begin{aligned} \max \quad & c_{true}^T x \\ \text{s.t.} \quad & x \in X_t. \end{aligned} \tag{1-3}$$

Note novamente que c_{true} nunca é revelado (ver Figura (1.1)).

O objetivo é que os estimadores $c^1, \ldots, c^T$ produzidos tenham a seguinte propriedade: na média, as ações $\bar{x}_t := \text{argmax}_{x \in X_t} \langle c_t, x \rangle$ obtidas otimizando as funções objetivo estimadas são próximas às soluções x_t que otimizam a função objetivo real c_{true} (por exemplo, queremos proximidade em termos da função objetivo real, ou seja, $\sum_t \langle c_{true}, \bar{x}_t \rangle \approx \sum_t \langle c_{true}, x_t \rangle$). (Note que estamos focando na versão do problema onde a função objetivo é linear).

For $t = 1, \ldots, T$
1. Algoritmo observa a região viável X_t
2. Algoritmo produz estimativas c_t
3. Algoritmo observa x_t

Figura 1.1: Visão geral do problema OInvOnline-FO

Como aplicação para o problema OInvOnline-FO voltamos novamente para o exemplo de aprendizado da função objetivo dos consumidores. Podemos assumir que existe um vetor de utilidades c_{true} dos produtos para um consumidor e esse vetor é desconhecido. Dado um horizonte de T dias, um mercado, e.g., a Amazon observa a cada dia $t \in T$ consumidores realizando suas compras. Primeiro ela observa o preço de seus produtos naquele dia, bem como as restrições de orçamento de um determinado consumidor. Depois ela faz uma estimativa da utilidade que cada um de seus produtos tem para tal consumidor, e só então a Amazon observa os produtos comprados por ele (sob a hipótese de que os consumidores maximizam suas utilidades).

O objetivo é aprender a preferência dos consumidores onde na média as soluções produzidas pela estimativa das utilidades, podemos interpretar como a recomendação de produtos para os consumidores, sejam tão boas quanto a solução ótima, ou seja, os produtos que são realmente comprados.

O problema OInvOnline-FO foi introduzido por Bärmann et al. em (32). No **caso específico que a função objetivo c_{true} é uma distribuição de probabilidade**, ou seja, $c_{true} \in [0, 1]$ para toda coordenada i e $\sum_i (c_{true})_i = 1$, eles propuseram um algoritmo baseado em *online learning* que produz estimativas com boas garantias:

Teorema 1 (Teorema 3.3 de (32)) *Para o problema* OInvOnline-FO*, o algoritmo de Bärmann et al. computa estimativas* $(c_1, \ldots, c_T)$ *com a seguinte garantia:*

$$\frac{1}{T}\left[\sum_{t=1}^{T} |\langle c_{true}, \bar{x}_t\rangle - \langle c_{true}, x_t\rangle|\right] \leq 2K\sqrt{\frac{\ln n}{T}} \tag{1-4}$$

onde $K \geq 0$ *é um limite superior do tamanho do maior vetor nas regiões viáveis* ($K \geq \max_{x_1, x_2 \in X_t} \|x_1 - x_2\|_\infty$) *e* $\bar{x}_t = \operatorname{argmax}_{x \in X_t} \langle c_t, x\rangle$ *é a solução ótima com relação às estimativas* $(c_1, ..., c_T)$ *calculadas.*

Note que essa taxa de erro média vai a zero conforme o horizonte de tempo aumenta.

OInvOnline-LD Um outro problema proposto por Bärmann et al. (32) é o aprendizado online das restrições do problema. Na verdade assume-se que as restrições são lineares, e deseja-se aprender o lado direito das restrições. O problema pode ser descrito da seguinte maneira: Existe um vetor b_{true} **desconhecido**, que é o lado direito das restrições. No início do processo o algoritmo observa a matriz de restrições A. Dado um horizonte de tempo T, no instante t, o algoritmo observa a função objetivo c_t. Nesse momento, ele tem que produzir uma estimativa $\bar{b}_t$ do lado direito b_{true}. Em seguida, o algoritmo obtém uma solução x_t, ótima com relação a b_{true}:

$$\begin{aligned} \max \quad & c_t^T x \\ \text{s.t.} \quad & Ax \leq b_{true} \\ & x \geq 0. \end{aligned} \tag{1-5}$$

Note que b_{true} nunca é revelado (ver Figura (1.2)).

O objetivo é que os estimadores $\bar{b}_1, \ldots, \bar{b}_T$ produzidos tenham a seguinte propriedade: Na média, as ações $\bar{x}_t := \text{argmax}\{\langle c_t, x\rangle \mid Ax \leq \bar{b}_t\}$ obtidas otimizando sob os lados direitos estimados são próximas ás soluções x_t que otimizam sob o lado direito real b_{true} (queremos proximidade em termos do valor objetivo real, dado pela otimização sob b_{true}, ou seja, $\sum_t \langle c_t, \bar{x}_t\rangle \approx \sum_t \langle c_t, x_t\rangle$). De maneira que $\bar{x}_t$ seja viável para o problema original, i.e., $\bar{x}_t \in \mathcal{X}$ para todo $t \in T$, onde $\mathcal{X}$ é a região viável do problema 1-5.

Entrada: matriz A
For $t = 1, \ldots, T$
1. Algoritmo observa a função objetivo c_t
2. Algoritmo produz estimativas $\bar{b}_t$
3. Algoritmo observa x_t

Figura 1.2: Visão geral do problema OInvOnline-LD

Apesar de terem proposto esse problema, Barmann et al. deixaram em aberto obter algoritmos com garantias provadas para ele.

1.2 Nossos Resultados

Nesta dissertação estudamos otimização inversa online, tanto na perspectiva teórica quanto na computacional, considerando os problemas OInvOnline-FO e OInvOnline-LD.

Contribuição 1: Na nossa primeira contribuição consideramos o problema OInvOnline-FO e generalizamos o resultado de Barmann et al. (32) para um vetor c_{true} qualquer, removendo assim a hipótese que esse vetor deveria ser uma distribuição de probabilidade.

Teorema 2 *Considere o problema* OInvOnline-FO. *O algoritmo* AlgoFO *computa estimativas* $(c_1, \ldots, c_T)$ *com a seguinte garantia:*

$$\frac{1}{T}\sum_{t=1}^{T} |\langle c_{true}, \bar{x}_t\rangle - \langle c_{true}, x_t\rangle| \leq \frac{8BD}{\sqrt{T}}, \tag{1-6}$$

onde $D = \|c_{true}\|_2$ *é a norma da função objetivo,* $B = \max_t \max_{x \in X_t} \|x\|_2$ *é o tamanho do maior vetor nas regiões viáveis e* $\bar{x}_t = \text{argmax}_{x \in X_t} \langle c_t, x \rangle$ *a solução ótima com relação às estimativas* $(c_1, ..., c_T)$ *calculadas .*

Notamos que independentemente, Barmann et al. em (33) obtiveram o mesmo resultado, mas com uma hipótese adicional que diâmetro máximo dos conjuntos viáveis X_t são conhecidos a priori.

Contribuição 2: Apresentamos uma solução para o problema OInvOnline-LD, deixado em aberto por Barmann et al. (32, 33).

Teorema 3 *Considere o problema* OInvOnline-LD. *O algoritmo* AlgoLD *produz estimativas* $\bar{b}_1, \ldots, \bar{b}_T$ *com as seguintes propriedades:*

1. *Para todo* t, $\bar{b}_t \leq b_{true}$ *(então toda solução viável para o problema com o lado direito* $\bar{x}_t$ *é viável para o problema original).*

2. *Essas soluções tem a seguinte garantia de valor:*

$$\frac{1}{T} \sum_{t=1}^{T} |\langle c_t, \bar{x}_t \rangle - \langle c_t, x_t \rangle| \leq \frac{2n^{3/2} mCB\Delta}{T}, \tag{1-7}$$

onde $\bar{x}_t = \text{argmax}\{\langle c_t, x \rangle \mid Ax \leq \bar{b}\}$ *é a solução ótima com relação às estimativas* $(\bar{b}^1, \ldots, \bar{b}^T)$ *calculadas,* $C = \max_t \|c_t\|_2$*, e* Δ *é o maior valor absoluto das matrizes* D^{-1}*, sendo* D *uma submatriz quadrada de* A *não singular.*

Contribuição 3: Realizamos também experimentos computacionais para demonstrar a aplicabilidade de nossa abordagem algorítmica bem como a sua eficácia. Investigamos seus usos em problemas de otimização combinatória clássicos como o problema da mochila (multidimensional), caminho mais curto com restrição de recurso, *lot sizing* e multifluxo.

2
Literatura em Otimização Inversa

Neste capítulo apresentaremos a literatura relacionada a otimização inversa. Apesar desse trabalho considerar apenas otimização inversa *online*, por completude discutimos nesse capítulo também otimização inversa não-online, que tem uma literatura muito mais rica. Os interessados somente nos problemas online podem pular diretamente para a Seção (2.6).

2.1
Otimização Inversa Offline em geral

Como discutido no Capítulo (1), dados a solução ótima conhecida $\hat{x}$, a estimativa $\hat{c}$ do parâmetro c, que parametriza $f(c, x)$ e a função de perda ϕ, o problema de otimização inversa é dado por,

$$\begin{aligned} \min \quad & \phi(c, \hat{c}) \\ \text{s.t.} \quad & f(c, \hat{x}) = \max_{x \in X} f(c, x). \end{aligned} \tag{2-1}$$

Na literatura encontramos o problema (2-1) com diferentes medidas para a função de perda $\phi(c, \hat{c})$. Os primeiros trabalhos utilizaram as normas ℓ_p, $\sum_{i=1}^{N}(c_i - \hat{c}_i)^p$, mais especificamente as normas ℓ_1, também conhecida como distância de Manhattan, $\sum_{i=1}^{N}|c_i - \hat{c}_i|$, a norma ℓ_2 ou Euclidiana, $\sum_{i=1}^{N}(c_i - \hat{c}_i)^2$, e a norma ℓ_∞, igualmente chamada de norma máxima ou infinita, $\max_{i=1,\ldots,N}(c_i - \hat{c}_i)$. A primeira medida utilizada foi a norma ℓ_2 (9) para o problema do caminho mais curto inverso. Posteriormente, passou-se a utilizar as normas ℓ_1 e infinita aplicadas a programação linear e otimização combinatória inversa, bem como a versão em pesos da norma ℓ_1 - $\sum_{i=1}^{N} w_i|c_i - \hat{c}_i|$. Conforme os estudos foram avançando, passou-se a considerar a distância de Hamming (HAM) como métrica, $\phi(c, \hat{c}) = \sum_{i=1}^{N} I_i(c, \hat{c})$, onde a função indicadora I_i é dada por,

$$I_i(c, \hat{c}) = \begin{cases} 0 & \text{if } c_i = \hat{c}_i, \\ 1 & \text{if } c_i \neq \hat{c}. \end{cases}$$

tal qual a sua versão em pesos, onde $\phi(c, \hat{c}) = w_i \sum_{i=1}^{N} I_i(c, \hat{c})$.

2.2 Otimização Inversa Linear

Depois dos trabalhos iniciais de Zhang et al. (11), que usaram a programação linear para resolver o problema do caminho mais curto inverso, estudos como o de Zhang e Liu (17) foram usados por Ahuja e Orlin (13) como base para a formulação de um *framework* padrão para otimização inversa linear. O *framework* proposto em (13), considera o problema de programação linear

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \geq 0 \end{aligned} \tag{2-2}$$

onde $c \in \mathbb{R}^n$ é o parâmetro que queremos inferir para função objetivo $f(c, x) = c^T x$, $A \in \mathbb{R}^{m \times n}$ é a matriz de restrições e $b \in \mathbb{R}^m$ é o lado direito. Dada uma solução viável $\hat{x}$ para o problema (2-2) com respeito a estimava $\hat{c}$ do parâmetro c, o problema inverso busca encontrar c que minimize uma função de perda ϕ. Eles utilizaram um método baseado na dualidade para resolver o problema e provaram que sob as funções de perda $\phi = \ell_1$ ou $\phi = \ell_\infty$, o problema inverso também pode ser escrito como um programa linear, resolvido em tempo polinomial.

Posteriormente, outros métodos de otimização inversa linear surgiram baseados no *framework* de Ajuha e Orlin. Lee (4) e Chan et al. (5) tratam o problema com um adicional de que a solução $\hat{x}$ não é necessariamente ótima para o problema (2-2) obtendo o que chamaram de generalização do problema de otimização inversa linear.

Trabalhos como o de Troutt et al.(7, 15), e Gallego e Javier (14) utilizaram versões dinâmicas da otimização inversa linear que também se basearam no framework de Ahuja e Orlin (13). Eles consideram um horizonte de tempo T e inferem o parâmetro c através da observação das soluções $x_t's$. Essa versão se diferencia do nosso problema online pois as informações não são reveladas em cada instante de tempo, mas são conhecidas no inicio do processo. Eles inferem o parâmetro c que na média minimiza o somatório dos gaps de dualidades obtidos por meio das soluções ótimas $x_1, \ldots, x_T$. Além do parâmetro c, Troutt et al. em (15) infere a matriz de restrições enquanto Gallego e Javier inferem o lado direito das restrições.

2.3 Otimização Combinatória Inversa

Otimização combinatória inversa é dado pelo problema (2-1), onde X é o conjunto de soluções do problema combinatório e $f(c, x)$ é linear, i.e.,$f(c, x) =$

$c^T x$. Historicamente, esses foram os primeiros estudos em otimização inversa. A maioria deles focam em transformar um um dado problema combinatório em outro que já possui um algoritmo bem estabelecido. Uma revisão extensiva da literatura de problemas de otimização combinatória inversa está disponível em (18). A seguir descreveremos os principais problemas de combinatórios inversos, suas aplicações e soluções.

2.3.1
Problema do caminho mais curto inverso

Dado um grafo $G = (V, E)$ e os vértices $s \in V$ e $t \in V$, respectivamente de origem e destino, o problema do caminho mais curto busca encontrar um caminho entre s e t a custo mínimo, dado o custo c_e de atravessar cada aresta $e \in E$ (portanto X representa o conjunto de todos os caminhos entre s e t). Já o problema inverso busca inferir os custos c_e das arestas, dado que conhecemos $\hat{x} \in X$, o caminho de menor custo entre s e t. As aplicações do problema do caminho mais curto inverso incluem modelagem de tráfego (2, 3), bem como tomografia sísmica e computadorizada (9–12) .

Burton e Toint em (9, 10) resolveram o problema do caminho mais curto inverso com a função de perda $\phi = \ell_2$ por meio de um algoritmo de programação quadrática. Como podemos modelar o conjunto viável de caminhos $s - t$ utilizando restrições lineares, o método de Ahuja e Orlin (13) mencionado na Seção (2.2), pode ser aplicado para se obter uma formulação linear desse problema inverso quando $\phi = \ell_1$ ou $\phi = \ell_\infty$. No caso em que ϕ é a norma ℓ_1 com pesos, Ahuja e Orlin (13) notou que o problema inverso pode ser resolvido usando problemas de fluxo.

Duin e Volgenant (35) consideraram o problema do caminho mais curto inverso com a função de perda $\phi = \text{HAM}$. Essa medida pode ser útil em problemas do mundo real, por exemplo, quando os custos só podem ser alterados dentro de um intervalo específico ou quando queremos minimizar o número de suas modificações. Duin e Volgenant mostram que resolver o problema é equivalente a resolver o *minimum cost circulation problem*, que pode ser resolvido em tempo polinomial via *minimum mean cycle-cancelling algorithm*.

2.3.2
Problema da Árvore geradora mínima inversa

Dado um grafo G conectado e não direcionado, o problema da árvore geradora mínima visa encontrar uma árvore que conecta todos os vértices do grafo G a custo mínimo, dado os custos c_e das arestas $e \in E$. O problema

inverso tem objetivo de inferir os custos das arestas c_e dado que conhecemos a árvore geradora mínima.

Sokkaligam et al. (36) mostraram que o problema inverso da árvore geradora mínima, com $\phi = \ell_1$, e sob determinadas condições, é análogo ao dual do problema da atribuição (*assignment*) em um grafo bipartido. Os autores desenvolveram um algoritmo eficiente baseado no *sucessive shortest path*. Eles também demostraram que quando ϕ é a norma ℓ_1 ponderada, o problema inverso é equivalente ao dual do problema de transporte. Além disso, os autores propuseram um algoritmo eficiente para o caso $\phi = \ell_\infty$. Em seguida, Ahuja e Orlin (16) expuseram que, para $\phi = \ell_\infty$, o dual da árvore geradora mínima inversa equivale ao *matching* bipartido ponderado em grafos estruturados, configuração esta usada por Hochbaum (23) para desenvolver um algoritmo mais eficiente para o problema.

O problema com $\phi = \text{HAM}$ foi estudado por (19, 22, 35). He e Zhang el al. (19, 22) mostram que resolver o problema equivale a resolver o *minimum-weight node cover problem*, em grafo bipartido, que reduzido ao problema de fluxo máximo, pode ser resolvido de maneira eficiente.

Por fim, Zhang et al. em (21) consideraram uma variação do problema, adicionando incerteza nos custos das arestas, onde esses custos desconhecidos não são fixos e podem ser alterados sob certas circunstancias não controladas. Uma aplicação seria a atualização dos custos de uma *Local Area Network* (LAN), dada pelos tempos de viagem. A incerteza vem do fato que a velocidade e o tempo de viagem dependem da largura da banda, que pode sofrer alterações. Eles derivaram dois modelos, cujo objetivo do primeiro é minimizar as soma das modificações nos custos e o segundo minimizar a modificação máxima, ambos considerando a incerteza. Zhang et al. mostraram que resolver os problemas sob incerteza é equivalente a resolver problemas determinísticos, que podem ser resolvidos pela otimização tradicional, quando as distribuições do problema inverso são conhecidas.

2.3.3 Problema da mochila inverso

Dada uma mochila de tamanho B e n itens, cada um com seu tamanho e valor, o problema da mochila consiste em determinar o subconjunto de itens com valor máximo, dentre o conjunto X de todos os subconjuntos de itens que cabem na mochila. Já o problema inverso, busca inferir o valor dos itens dado que conhecemos o subconjunto que maximiza o valor da mochila, e o conjunto X.

Huang (27) mostrou que o problema da mochila inverso, quando $\phi =$

ℓ_2 e o peso de cada item é inteiro, pode ser resolvido em tempo pseudo polinomial. Para isso, o autor construiu um grafo direcionado, de forma que, para solucionar o problema da mochila inverso basta resolver o problema do caminho mais curto inverso nesse grafo. Ou seja, pode-se utilizar as mesmas abordagens descritas na Seção 2.3.1 .

Buscando solucionar o problema de forma geral, Rolland et al. (37) propuseram dois modelos, onde, no primeiro, $\phi = \ell_1$ e outro quando $\phi = \ell_\infty$, esses modelos são reduzidos a um programa linear inteiro. Eles provam que os problemas são respectivamente co-NP-Completo e co-NP-Difícil. Por fim, desenvolveram uma formulação bi-nível, para os dois modelos, que pode ser resolvida por algoritmos de *branch-and-bound* ou *branch-and-cut*.

2.4 Otimização Inversa Cônica/Convexa

O problema mais geral de otimização inversa cônica foi estudada por Iyegar em (1). As aplicações desse método incluem otimização de portfólio e identificação de funções utilidade. Seguindo as convenções de (1), o problema de programação cônica tem a seguinte forma

$$\begin{aligned} \max \quad & f(x,c) \\ \text{s.t.} \quad & g(x) \in \mathcal{K} \\ & Ax = b, \end{aligned} \tag{2-3}$$

onde $f(x,c) : \mathbb{R}^n \to \mathbb{R}$ é convexa e diferenciável em x, $g : \mathbb{R}^n \to \mathbb{R}^s$ é uma função de valor vetorial côncava diferenciável e $\mathcal{K} \in \mathbb{R}^m$ é um cone. (veja (1) para definições dessas noções.)

O problema inverso é então encontrar o parâmetro c dada uma solução ótima para o problema, minimizando a função de perda $\phi = \ell_p$ para $p \geq 1$. Iyegar utilizou um método baseado nas condições de KKT para resolver o problema.

Por fim, Aswani et. al (26) consideraram a otimização inversa convexa sob o contexto de inferência estatística. Onde o problema de programação convexa é dado por

$$\begin{aligned} \min \quad & f(x,u,c) \\ \text{s.t.} \quad & g(x,u,c) \leq 0, \end{aligned} \tag{2-4}$$

onde, $x \in \mathbb{R}^d$ é a variável de decisão, $u \in \mathbb{R}^m$ uma variável de entrada externa, $c \in \mathbb{R}^p$ o vetor de parâmetros. $f : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ e $g : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^q$ são funções contínuas e convexas em x para u e c fixos. Considere os conjuntos $\mathcal{S}(u,c) = \text{argmin}\{f(x,u,c)|g(x,u,c) \leq 0\}$, $\mathcal{V}(u,c) = \min\{f(x,u,c)|g(x,u,c) \leq 0\}$ e $\Phi(u,c) = \{x \in \mathbb{R}^d | g(x,u,c) \leq 0\}$, que

representam respectivamente conjunto solução, conjunto de valores da função objetivo e conjunto de soluções viáveis, todos relacionados ao problema 2-4.

Assim, o problema de otimização inversa convexo pode ser descrito como,

$$\begin{aligned} \min \quad & Q(\theta) \\ \text{s.t.} \quad & \theta \in \Theta, \end{aligned} \tag{2-5}$$

onde $\Theta \subseteq \mathbb{R}^p$ é um conjunto convexo conhecido e

$$Q(\theta) = \mathbb{E}\left(\min_{x \in \mathcal{S}(u,\theta)} ||y - x||^2 \right) \tag{2-6}$$

tal que u e y são variáveis aleatórias distribuídas sobre alguma distribuição $\mathbb{P}_{(u,y)}$ conjunta desconhecida, onde $\mathcal{U} \times Y \subseteq \mathbb{R}^m$ é o seu suporte. Como $\mathbb{P}_{(u,y)}$ é desconhecida eles mostraram que podemos resolver de forma aproximada o problema inverso usando o método *Sample Average Approximation* (SAA).

Além disso, Aswani et.al (26) provaram que se o problema de otimização, bem como o seu inverso são convexos, então resolver o problema inverso é NP-hard. Porém sob algumas condições eles podem ser solucionados em tempo polinomial.

2.5 Otimização Inversa Inteira

Existem poucos estudos relacionados a otimização inversa inteira, Huang (27) foi primeiro a tratar do problema, com um número fixo de restrições. Considere problema de programação inteira

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \in \mathbb{Z}^n \end{aligned} \tag{2-7}$$

onde a matriz de restrições $A \in \mathbb{Z}^{m \times n}$, o vetor do lado direito das restrições $b \in \mathbb{Z}^m$ e e a função objetivo $c \in \mathbb{Z}^n$ são inteiros. Huang (27) construiu um grafo G, representando o problema (2-7) e então demostrou que uma solução viável para problema inteiro equivale a um caminho em G, para um determinado vértice de origem e destino. Portanto, resolver o problema de otimização inversa inteiro, equivale a resolver ao problema do caminho mais curto no grafo direcionado G, quando $\phi = \ell_p$, para $p = \{1, 2, \infty\}$.

Schaefer (28) propôs um modelo para o otimização inversa para problemas inteiros gerais, ele o formulou baseado na dualidade de programação inteira superaditiva, onde eles consideram um programa inteiro similar ao (2-7), cuja a função objetivo $c \in \mathbb{R}^n_+$. Quando ϕ é a norma ℓ_1, o objetivo do problema

inverso é encontrar $c \in C^*$, onde C^* sumariza o conjunto viável do problema inverso, dado por

$$\min_{c \in C^*} \sum_{i=1}^{n} |\hat{c}_i - c_i| \tag{2-8}$$

no qual $\hat{c}$ é conhecido. Schaefer(28) desenvolveu duas abordagens para resolver o problema inverso inteiro. A primeira, consiste em encontrar o casco convexo do problema inteiro (2-7), com $c \in \mathbb{R}^n_+$ e aplicar os métodos de otimização inversa linear pra resolvê-lo. Já a segunda abordagem consiste em reduzir o problema de otimização inversa inteiro em um programa linear. Nenhuma das duas abordagens tem boa performance para grandes instâncias.

Considere agora o problema de otimização inversa linear inteiro misto com ϕ igual a norma ℓ_1 ponderada,

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i^T |c_i - \hat{c}_i| \\ \text{s.t.} \quad & \hat{x} = \text{argmax}\{c^T x : Ax \leq b, x \geq 0, x_l \in \mathbb{Z}\}. \end{aligned} \tag{2-9}$$

Wang (29) apresentou um algoritmo de plano de cortes para o problema inverso (2-9). Ele resolve o problema sob a norma ℓ_∞ de maneira similar. Em seguida, Duan e Zhang (30) desenvolveram uma heurística para aprimorar o algoritmo de (29).

2.6 Otimização Inversa Online

Voltamos agora para otimização inversa online, o foco do nosso trabalho. Relembramos os problemas OInvOnline-FO e OInvOnline-LD.

OInvOnline-FO: Relembramos brevemente a definição do problema OInvOnline-FO. Considere um vetor desconhecido c_{true} que parametriza a função objetivo. No tempo t, o algoritmo observa a região viável X_t e produz uma estimativa c_t da função objetivo c_{true}. Em seguida, obtém uma solução $x_t \in X_t$ ótima com relação a c_{true}:

$$\begin{aligned} \max \quad & c_{true}^T x \\ \text{s.t.} \quad & x \in X_t. \end{aligned} \tag{2-10}$$

O objetivo, diferente do problema de otimização inversa offline, é que os estimadores $c^1, \ldots, c^T$ produzidos tenham a seguinte propriedade: Na média, as ações $\bar{x}_t := \text{argmax}_{x \in X_t} \langle c_t, x \rangle$ obtidas otimizando as funções objetivo estimadas são próximas às soluções x_t que otimizam a função objetivo real c_{true}, i.e., queremos proximidade em termos da função objetivo real ($\sum_t \langle c_{true}, \bar{x}_t \rangle \approx \sum_t \langle c_{true}, x_t \rangle$).

Como mencionando anteriormente, Bärmann et al. (32, 33) propuseram o problema OInvOnline-FO e desenvolveram um algoritmo baseado no método *Multiplicative Weights Update* (MWU), que será descrito na Seção (3.1).

Teorema 4 (Teorema 3.3 de (32)) *Considere o problema* OInvOnline-FO. *Assuma que c_{true} é uma distribuição de probabilidade. Então existe um algoritmo que computa as estimativas $(c_1, \ldots, c_T)$ com a seguinte garantia:*

$$\frac{1}{T}\left[\sum_{t=1}^{T} |\langle c_{true}, \bar{x}_t\rangle - \langle c_{true}, x_t\rangle|\right] \leq 2K\sqrt{\frac{\ln n}{T}} \tag{2-11}$$

onde K é uma cota superior para o diâmetro dos conjuntos X_t ($K \geq \max_{x_1,x_2\in X_t} \|x_1 - x_2\|_\infty$), e $\bar{x}_t = \operatorname{argmax}_{x\in X_t}\langle c_t, x\rangle$ é a solução ótima com relação às estimativas $(c_1, ..., c_T)$ calculadas.

Subsequentemente, independente do nosso trabalho, Bärmann et al. (33) generalizaram sua estratégia anterior removendo a hipótese de que c_{true} é uma distribuição de probabilidade.

Teorema 5 (Teorema 3.10 de (33)) *Considere o problema* OInvOnline-FO. *Existe uma algoritmo que computa as estimativas $(c_1, \ldots, c_T)$ com a seguinte garantia:*

$$\frac{1}{T}\left[\sum_{t=1}^{T} |\langle c_{true}, \bar{x}_t\rangle - \langle c_{true}, x_t\rangle|\right] \leq \frac{3LB}{2\sqrt{T}} \tag{2-12}$$

onde L é o diâmetro do conjunto de pontos projetados pelo OGD e B é o diâmetro de da região viável X_t ($B = \max_t \max_{x\in X_t} \|x\|_2$) e $\bar{x}_t = \operatorname{argmax}_{x\in X_t}\langle c_t, x\rangle$ é a solução ótima com relação às estimativas $(c_1, ..., c_T)$ calculadas .

Esse resultado é muito semelhante ao nosso primeiro resultado desta dissertação (Teorema (2)). Entretanto, para obter a garantia do Teorema 5, Bärmann et al. assumiu conhecer a priori o diâmetro da região viável X_t, que não é conhecida no início do processo, dado que X_t é exibido de forma online. Nossa abordagem funciona para o caso onde não conhecemos o diâmetro de X_t.

OInvOnline-LD: Relembramos brevemente a definição do problema OInvOnline-LD. Existe um vetor b_{true} desconhecido, que é o lado direito das restrições (assume-se que as restrições são lineares). No início do processo o algoritmo observa a matriz de restrições A. Dado um horizonte de tempo T, no instante t, o algoritmo observa a função objetivo c_t e tem que produzir uma estimativa $\bar{b}_t$ do lado direito b_{true}. Em seguida, o algoritmo obtém uma solução x_t, ótima com relação a b_{true}:

$$\begin{aligned} \max \quad & c_t^T x \\ \text{s.t.} \quad & Ax \leq b_{true} \\ & x \geq 0 \end{aligned} \tag{2-13}$$

O objetivo é que os estimadores $\bar{b}^1, \ldots, \bar{b}^T$ produzidos tenham a seguinte propriedade: Na média, as ações $\bar{x}_t := \text{argmax}\{\langle c_t, x\rangle \mid Ax \leq \bar{b}_t\}$ obtidas otimizando sob os lados direitos estimados são próximas ás soluções x_t que otimizam sob o lado direito real b_{true}, i.e., queremos proximidade em termos do valor objetivo real, dado pela otimização sob b_{true} ($\sum_t \langle c_t, \bar{x}_t\rangle \approx \sum_t \langle c_t, x_t\rangle$). De maneira que $\bar{x}_t$ seja viável para o problema original, ou seja, $\bar{x}_t \in \mathcal{X}$ para todo $t \in T$, onde $\mathcal{X}$ é a região viável do problema (2-13).

Esse problema foi apenas proposto por Bärmann et al. em (32) e deixado como questão em aberto obter um algoritmo com garantias para esse problema. Demonstramos como aprender as restrições através de um algoritmo online detalhado no Capítulo 5.

Versão Generalizada da Otimização Inversa Online: Este problema foi proposto por Dong et.al em (34) e consiste no aprendizado online de parâmetros do problema. Esses parâmetros são dados em termos da função objetivo e restrições, onde ambos são descritos por funções convexas. O problema pode ser descrito da seguinte maneira: existe um parâmetro θ_{true} desconhecido. Dado um horizonte de tempo T, no instante t, o algoritmo observa u_t, onde u_t é um parâmetro conhecido do problema e produz uma estimativa θ_t do parâmetro θ_{true} e uma solução $y \in \mathcal{Y}$ com respeito a essa estimativa, que não é necessariamente ótima com respeito a θ_{true}. Em seguida, o algoritmo obtém uma solução x_t ótima com respeito a θ_{true}:

$$\begin{aligned} \min \quad & f(x, u, \theta_{true}) \\ \text{s.t.} \quad & g(x, u, \theta_{true}) \leq 0. \end{aligned} \tag{2-14}$$

e assim o algoritmo sofre a perda $\ell_{(y_t, u_t, \theta_t)} = \min_{x \in S(u,\theta)} \|y - x\|_2^2$, onde $S(u, \theta)$ é o conjunto de soluções ótimas do problema (2-14). O objetivo é aprender o parâmetro θ_{true} do problema (2-14), minimizando o seu regret, que é dado pela perda cumulativa do algoritmo em todos os instantes de tempo T contra o θ que minimiza a perda para todo t,

$$\sum_{t=1}^{T} \ell(y_t, u_t, \theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^{T} \ell(y_t, u_t, \theta)$$

Dong et.al em (34) desenvolveram um algoritmo online que aprende o parâmetro θ_{true} assumindo que as funções de perda são uniformemente $\frac{4(B+R)k}{\lambda}$-

Lipschitz e que $f(x, u, \theta)$ é λ-fortemente convexa. O Teorema a seguir fornece a garantia precisa.

Teorema 6 (Teorema 3.2 de (34)) *Considere a versão generalizada da otimização inversa online para aprender a função objetivo. Existe um algoritmo que computa estimativas $(\theta_1, \ldots, \theta_T)$ com a seguinte garantia:*

$$\sum_{t=1}^{T} \ell(y_t, u_t, \theta_t) - \min_{\theta \in \Theta} \sum_{t=1}^{T} \ell(y_t, u_t, \theta) \leq \frac{4\sqrt{2}(B+R)Dk}{\lambda}\sqrt{T}. \tag{2-15}$$

onde Θ é um conjunto convexo compacto, $D \geq \|\theta_{true}\|_2$, B é o limite superior das regiões viáveis do problema, R é o limite superior das distâncias dos valores de y_t em relação ao conjunto de soluções ótimas $S(u, \theta)$ e $k > 0$ é a constante Lipschitz.

Os resultados obtidos por Dong et.al em (34), de forma independente, generalizam o nosso no caso de aprender a função objetivo, já que eles aprendem funções objetivo convexas. Quanto ao aprendizado das restrições eles apenas dizem como aprender porém não demonstram.

3
Preliminares: *Online Learning*

Como mencionado na introdução, um dos nossos algoritmos para o problema de otimização inversa online utilizará como base algoritmos de *Online Learning*, descritos nesse capítulo.

Online Learning (38, 39) é um modelo de otimização sequencial, que pode ser pensado como um jogo entre um jogador (i.e., aprendiz ou algoritmo de aprendizagem) e um adversário. Dado um conjunto de possíveis ações $\mathcal{A}$, o jogo se desenrola em T rodadas, onde em cada uma o jogador e o adversário seguem o seguinte protocolo:

1. Jogador escolhe uma ação $a_t \in \mathcal{A}$
2. Adversário escolhe uma função de perda $f_t : \mathcal{A} \to \mathbb{R}$
3. Jogador sofre a perda $f_t(a_t)$ associada a ação a_t escolhida;
4. Jogador observa a função de perda f_t.

Portanto, as funções de perda de cada instante de tempo são reveladas uma-a-uma, e no tempo t o algoritmo toma sua decisão somente com a informação das funções $f_1, \ldots, f_{t-1}$ vistas ate o tempo anterior. O objetivo é minimizar o *regret*, que se refere a perda cumulativa do jogador em relação a perda cumulativa da melhor ação fixa em retrospectiva:

$$\text{regret} := \sum_{t=1}^{T} f_t(a_t) - \min_{a \in \mathcal{A}} \sum_{t=1}^{T} f_t(a). \tag{3-1}$$

Uma propriedade importante desse modelo é que ele não assume nenhuma hipótese sobre as funções de perda, e.g., que elas tem comportamento estocástico. Portanto, estamos interessado em cotas superiores para o regret de pior-caso, ou seja, para todas as instancias de uma classe (e.g., funções convexas e Lipschitz). Essa é uma propriedade fundamental que deixa esse modelo ser utilizado em diversos contextos, e em geral como uma peça de um sistema que interage com outras peças complexas: as garantias valem independente desse comportamento complexo.

Online Linear Optimization (OLO) é um caso especial do problema quando o conjunto viável $\mathcal{A}$ é convexo e as funções de perda f_t são lineares.

Para ilustrar, descrevemos um caso particular de OLO, o clássico problema de *previsão com dicas de experts* (EXPERTS). A idéia é que existem m experts disponíveis que em cada instante de tempo dizem a sua recomendação para o tempo corrente (e.g., a previsão do movimento de um ação na bolsa de valores). O objetivo é, sem saber a priori a qualidade dos experts, tomar decisões cuja perda é comparável com a do melhor expert. Formalmente, no tempo t o algoritmo escolhe uma distribuição de probabilidade $p^t = (p_1^t, ..., p_m^t)$ sobre os m experts. Em seguida, o vetor de perdas $\ell^t = (\ell_1^t, ..., \ell_m^t)$ indicando a perda de cada expert é revelado, e o algoritmo sofre perda $\langle p^t, \ell^t \rangle$ associada à distribuição p^t escolhida sobre os experts (que pode ser interpretada como o valor esperado da perda incorrida caso um experts seja escolhido de forma aleatória com distribuição p^t). O objetivo é ter perda total comparável com a do melhor expert, ou seja, queremos

$$\sum_{t=1}^{T} \langle p^t, \ell^t \rangle \lesssim \min_{i \in [m]} \sum_{t=1}^{T} \ell_i^t.$$

Note que esse problema corresponde precisamente ao problema OLO com funções de perda $f_t(p) := \langle p, \ell^t \rangle$ e conjunto viável $\mathcal{A}$ sendo as distribuições sobre m item.

Mesmo esse caso especial do problema OLO tem diversas aplicações, incluindo resolução rápida de problemas lineares do tipo *packing/covering*, *Boosting*, e, mais de forma mais geral, resolução de forma aproximada de problemas minimax; veja (40) para essas e muitas outras aplicações.

A seguir descreveremos os algoritmos clássicos com garantias de baixo *regret* para os problemas EXPERTS e OLO.

3.1 *Multiplicative Weights Update*

O *Multiplicative Weights Update* (MWU) é um dos algoritmos fundamentais para o problema EXPERTS. A idéia em alto-nível é atribuir pesos aos experts que são atualizados conforme suas performance, penalizando mais os experts com maior perda; crucialmente essa atualização é feita de forma multiplicativa. A distribuição sobre os experts é então obtida normalizando esses pesos.

Mais formalmente, o algoritmo tem a seguinte descrição:

Algorithm 1 MWU

1: **Entrada:** Taxa de aprendizado $\eta > 0$
2: $w_i^1 \leftarrow 1 \quad \forall i \in \{1, ..., m\}$
3: **for** `t = 1 to T` **do**
4: Joga a distribuição p^t dada por $p_i^t := \frac{w_i^t}{\sum_j w_j^t}$
5: Observa o vetor de perdas ℓ^t
6: Sofre a perda $\langle p^t, \ell^t \rangle$
7: Atualiza os pesos $w_i^{t+1} \leftarrow w_i^t(1 - \eta \ell_i^t)$ para todo $i \in \{1, \ldots, m\}$
8: **end for**

Apesar de sua simplicidade, esse algoritmo possui uma excelente garantia de regret de ordem $O(\sqrt{T})$. Isso significa que o regret médio por iteração é $O(\frac{1}{\sqrt{T}})$, indo para zero conforme o horizonte T do jogo aumenta. O teorema seguinte fornece a garantia precisa.

Teorema 7 (Teorema 2.1 de (40)) *Suponha que as perdas dos experts ℓ^t pertencem ao conjunto $[0, 1]^m$. Então o algoritmo MWU com $\eta = \sqrt{\frac{\log m}{T}}$ tem a seguinte garantia:*

$$\sum_{t=1}^{T} \langle \ell^t, p^t \rangle \leq \min_{i \in [m]} \sum_{t=1}^{T} \ell_i^t + 2\sqrt{T \log m}. \tag{3-2}$$

Além disso, é sabido que essa garantia é essencialmente ótima, mesmo quando as perdas ℓ^t são estocásticas e amostradas de uma distribuição conhecida (40).

3.2 *Follow the Regularized Leader*

Passando para o problema mais geral OLO, um dos algoritmos fundamentais para esse modelo é o chamado *Follow the Regularized Leader* (FTRL). Em alto-nível ele é um procedimento guloso regularizado. Lembre que no problema OLO, no tempo t o algoritmo tem informação das funções de perdas passadas $f_1, \ldots, f_{t-1}$ e precisa produzir um ponto $a_t \in \mathcal{A}$. Nesse momento, o algoritmo FTRL escolhe a ação que minimiza a soma dessa perdas anteriores mais um termo de regularização $R : \mathcal{A} \to \mathbb{R}$ fortemente convexo:

Algorithm 2 Follow the Regularized Leader

1: **Entrada:** Função fortemente convexa R, e taxa de aprendizado $\eta > 0$
2: **for** `t = 1 to T` **do**
3: Joga $a_t := \text{argmin}_{a \in \mathcal{A}} \left\{ \sum_{i=1}^{t-1} f_i(a) + \frac{1}{\eta} R(a) \right\}$
4: Observa a função de perda f_t
5: Sofre perda $f_t(a_t)$
6: **end for**

Observamos que o objetivo da regularização adicionada é evitar *overfitting* do passado. Apesar de existirem várias funções de regularização importantes, como por exemplo a entropia negativa,[1] nessa dissertação consideraremos somente a regularização Euclideana $R(a) = \frac{1}{2}\|a\|_2^2$. Nesse caso, é sabido que o algoritmo FTRL é equivalente ao algoritmo *Online Gradient Descent* de Zinkevich (41), veja Seção 5.4.1 de (39).

Mesmo nesse problema OLO mais geral, o algoritmo FTRL, com taxa de aprendizado setada corretamente, também possui garantia de regret $O(\sqrt{T})$. Mais precisamente, temos a seguinte garantia em função da taxa de aprendizado η (na nossa aplicação, utilizaremos FTRL com diferentes η's, e essa flexibilidade será importante para nossas garantias).

Teorema 8 (Teorema 5.1 de (39)) *Considere uma instância do problema OLO com funções lineares $f_t(a) = \langle \ell^t, a \rangle$. Quando executado sobre essa instância, o algoritmo FTRL com regularização $R(a) = \frac{1}{2}\|a\|_2^2$ possui a seguinte garantia (para todo $\eta > 0$):*

$$\sum_{t=1}^{T} f_t(a_t) \leq \min_{a \in \mathcal{A}} \sum_{t=1}^{T} f_t(a) + 2\eta T B^2 + \frac{D^2}{2\eta}, \tag{3-3}$$

onde $B := \max_t \|\ell^t\|_2$ é o maior tamanho dos gradientes das perdas, e $D := \max_{a \in \mathcal{A}} \|a\|_2$ (essencialmente o diâmetro de $\mathcal{A}$).

[1] Pode-se verificar que o algoritmo MWU da seção anterior é precisamente o algoritmo FTRL com regularização neg-entropica, i.e., $R(a) = \sum_i p_i \log p_i$.

4
Aprender a Função Objetivo

Neste capítulo consideramos o problema OInvOnline-FO de aprender a função objetivo c_{true} do problema

$$\begin{aligned} \max \quad & c_{true}^T x \\ \text{s.t.} \quad & x \in X_t. \end{aligned} \tag{4-1}$$

a partir de observações dos X_t's e soluções ótimas x_t's (veja Seção (1.1) para relembrar a definição completa do problema).

A nossa primeira contribuição é o seguinte resultado.

Teorema 9 *O algoritmo* AlgoFO *computa vetores c_t (estimativas de c_{true}) com a seguinte propriedade: sendo $\bar{x}_t = \text{argmax}_{x \in X_t} \langle c_t, x \rangle$ a solução ótima com relação às estimativas calculadas,*

$$\frac{1}{T} \sum_{t=1}^{T} \langle c_t - c_{true}, \bar{x}_t - x_t \rangle \leq \frac{8BD}{\sqrt{T}}, \tag{4-2}$$

onde $D = \|c_{true}\|_2$ é a norma da função objetivo, e $B = \max_t \max_{x \in X_t} \|x\|_2$ é o tamanho do maior vetor nas regiões viáveis.

Note que esse garantia implica a garantia

$$\frac{1}{T} \sum_{t=1}^{T} |\langle c_{true}, x_t \rangle - \langle c_{true}, \bar{x}_t \rangle| \leq \frac{8BD}{\sqrt{T}}$$

enunciada no Teorema 2: utilizando a otimalidade de $\bar{x}_t$ com relação a c_t e de x_t com relação a c_{true}, temos:

$$\begin{aligned} \frac{8BD}{\sqrt{T}} \geq \frac{1}{T} \sum_{t=1}^{T} \langle c_t - c_{true}, \bar{x}_t - x_t \rangle &= \frac{1}{T} \sum_{t=1}^{T} \left[\underbrace{\langle c_t, \bar{x}_t - x_t \rangle}_{\geq 0} + \langle c_{true}, x_t - \bar{x}_t \rangle \right] \\ &\geq \frac{1}{T} \sum_{t=1}^{T} \underbrace{\langle c_{true}, x_t - \bar{x}_t \rangle}_{\geq 0} \\ &= \frac{1}{T} \sum_{t=1}^{T} |\langle c_{true}, x_t \rangle - \langle c_{true}, \bar{x}_t \rangle|. \end{aligned}$$

4.1
Algoritmo inicial: assumindo diâmetros conhecidos

Para simplificar a exposição, começamos descrevendo e analisando um algoritmo que assume conhecimento do "diâmetro" máximo dos conjuntos viáveis X_t, ou seja, definido $diam(X) := \max_{x \in X} \|x\|_2$, assumimos conhecimento de $B := \max_t diam(X_t)$. Note que $diam(X)$ não é exatamente o diâmetro do conjunto X, mas utilizamos esse mnemonico pois é de fácil visualização (na verdade poderíamos assumir, translacionando os conjuntos, sem perda de generalidade que todos os X_t's contém a origem, e nesse caso $2 \cdot diam(X_t)$ é uma cota superior no diâmetro de X_t).

Note que B **não é conhecido** de antemão, pois depende dos conjuntos viáveis X_t que só são exibidos online. Porém, assumindo que somos munidos desse parâmetro B, o algoritmo AlgoDiamFO é o seguinte: (utilizamos FTRLm(η) para denotar o algoritmo FTRL com regularizador Euclideano $\frac{1}{2}\|\cdot\|_2$, taxa de aprendizado η, e conjunto viável $\mathcal{A} = \{c \in \mathbb{R}^m : \|c\|_2 \leq D\}$, onde $D = \|c_{true}\|_2$)

Algorithm 3 AlgoDiamFO

Entrada: T, $D = \|c_{true}\|_2$, $B = \max_t diam(X_t)$

$\eta \leftarrow \frac{D}{4B\sqrt{T}}$
Inicializa FTRLm(η)
for $t = 1$ to T **do**
    Observa a região viável X_t
    Define a estimativa c_t como o FTRLm(η) nessa rodada
    $\bar{x}_t \leftarrow \text{argmax}_{x \in X_t} \langle c_t, x \rangle$
    Observa a solução ótima x_t ($\text{argmax}_{x \in X_t} \langle c_{true}, x \rangle$)
    Define o vetor de perdas $\ell_t \leftarrow \bar{x}_t - x_t$
    Passa a função de perda $f_t(\cdot) := \langle \ell_t, \cdot \rangle$ para FTRLm(η), para sua atualização
end for

Note que o conhecimento de B é utilizado para setar a taxa de aprendizado η. Lembramos que quando o algoritmo FTRL utiliza o regularizador Euclideano $\frac{1}{2}\|\cdot\|_2$ ele é equivalente ao algoritmo *Online Gradient Descent* (OGD), algoritmo online utilizado em (33) para aprender $c'_{true}s$ mais gerais.

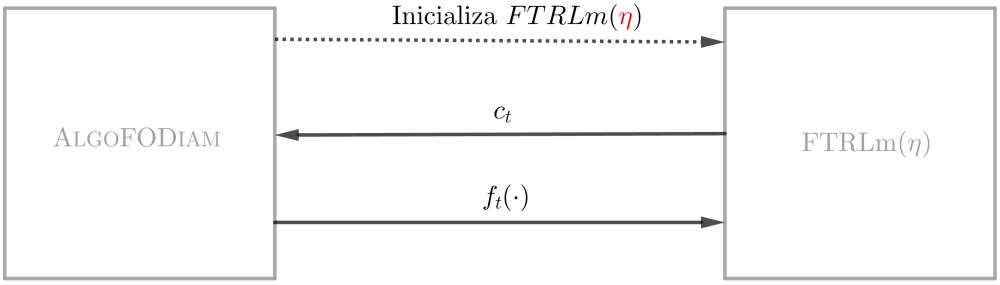


Figura 4.1: Iteração do AlgoDiamFO com o FTRL

Lema 1 *O algoritmo* AlgoDiamFO *satisfaz as garantias do Teorema 9.*

Prova. Para aplicarmos a garantia do FTRL do Teorema 8 a FTRLm(η) precisamos cotas superiores para $\max_{c\in\mathcal{A}} \|c\|_2$ e $\|\ell_t\|_2$. Por definição de $\mathcal{A}$, o primeiro termo é limitado por cima por $D = \|c_{true}\|_2$. Para o segundo termo, pela desigualdade triangular temos

$$\|\ell_t\|_2 = \|\bar{x}_t - x_t\|_2 \leq \|\bar{x}_t\|_2 + \|x_t\|_2 \leq 2B, \tag{4-3}$$

onde na última desigualdade utilizamos que $\bar{x}_t, x_t \in X_t$ e a definição de B. Portanto, aplicando o Teorema 8 a FTRLm(η) obtemos

$$\begin{aligned}
\sum_{t=1}^{T}\langle \bar{x}_t - x_t, c_t - c_{true}\rangle &= \sum_{t=1}^{T}\langle \ell_t, c_t - c_{true}\rangle \\
&= \sum_{t=1}^{T}\Big(f_t(c_t) - f_t(c_{true})\Big) \\
&\leq \sum_{t=1}^{T}\Big(f_t(c_t) - \min_{c\in\mathcal{A}} f_t(c)\Big) \qquad (4\text{-}4)\\
&\leq 8\eta T B^2 + \frac{D^2}{2\eta} \qquad (4\text{-}5)\\
&= 4BD\sqrt{T},
\end{aligned}$$

onde na última igualdade utilizamos a definição de η setada no algoritmo. Dividindo ambos os lados por T obtemos o resultado desejado. ■

4.2 Algoritmo AlgoFO

Como mencionado anteriormente, não sabemos de antemão o parâmetro $B = \max_t diam(X_t)$ e portanto não podemos setar a taxa de aprendizado η de forma ótima como feita no algoritmo AlgoDiamFO. Para contornar

isso, utilizamos o chamado "doubling trick": mantemos uma estimativa $\tilde{B}$ (valor pequeno) do parâmetro B; ao notarmos que a estimativa $\tilde{B}$ estava pequena demais, ou seja, ao encontrarmos uma iteração com $\tilde{B} < diam(X_t)$, essencialmente dobramos o valor de $\tilde{B}$. Ou seja, o algoritmo procede em *fases*, cada fase possuindo seu valor de $\tilde{B}$. Em cada uma dessas fases o algoritmo reinicia FTRLm e seta sua taxa de aprendizado η como em AlgoDiamFO mas utilizando a estimativa atual $\tilde{B}$: $\eta = \frac{D}{4\tilde{B}\sqrt{T}}$. A descrição mais formal do algoritmo AlgoFO é a seguinte:

Algorithm 4 AlgoFO

Entrada: T, $D = \|c_{true}\|_2$

$i \leftarrow 1$ # número da fase atual do algoritmo
$\tilde{B}_1 = 1$ # estimativa de B dessa fase
$\eta_1 \leftarrow \frac{D}{4\tilde{B}_i\sqrt{T}}$ # taxa de aprendizado dessa fase
Inicializa FTRLm(η_1)
for $t = 1$ to T **do**
    Observa a região viável X_t
    if $diam(X_t) > \tilde{B}_i$ **then** # inicia nova fase
        $i \leftarrow i + 1$
        $\tilde{B}_i = \max\{2\tilde{B}_{i-1}, diam(X_t)\}$
        $\eta_i \leftarrow \frac{D}{4\tilde{B}_i\sqrt{T}}$
        Reinicializa FTRLm(η_i)
    end if
    Define a estimativa c_t como o FTRLm(η_i)
    $\bar{x}_t \leftarrow \text{argmax}_{x \in X_t} \langle c_t, x \rangle$
    Observa a solução ótima x_t ($\text{argmax}_{x \in X_t} \langle c_{true}, x \rangle$)
    Define o vetor de perdas $\ell_t \leftarrow \bar{x}_t - x_t$
    Passa a função de perda $f_t(\cdot) := \langle \ell_t, \cdot \rangle$ para FTRLm(η_i)
end for

A intuição para a análise do algoritmo é que podemos aplicar essencialmente a análise do algoritmo AlgoDiamFO a cada fase, e somar sobre todas as fases. O fato de dobrarmos a estimativa de B em cada fase ajuda a "formar uma progressão geométrica" onde a "soma dos erros" será absorvida pelo último termo. Provaremos agora formalmente a garantia desse algoritmo.

Prova.[Demonstração do Teorema 9] Define $fase_k \subseteq [T]$ como o conjunto de iterações t da fase k, ou seja, onde temos $i = k$. Devido ao reinicio do FTRLm em cada fase, temos que na fase k, os pontos $(c_t)_{t \in fase_k}$ são produzidos pelo algoritmo FTRL(η_k) executado sobre as funções de perdas $(f_t)_{t \in fase_k}$. Portanto, podemos aplicar a garantia do FTRL dentro dessa fase. Como na prova do

Lema 1, temos a cota superior $\max_{c\in\mathcal{A}} \|c\|_2 \leq D$, por definição de $\mathcal{A}$. Como em (4-3), temos que pra todo t na fase k

$$\|\ell_t\|_2 = \|\bar{x}_t - x_t\|_2 \leq \|\bar{x}\|_2 + \|x_t\|_2 \leq 2diam(X_t),$$

onde a última desigualdade utiliza o fato que $\bar{x}_t, x_t \in X_t$. Porém, note que $\tilde{B}_k$ é uma cota superior para $diam(X_t)$ para todos os tempos t na fase k: note que isso vale para o primeiro tempo $t \in fase_k$, pois nesse caso por definição $\tilde{B}_k = \max\{2\tilde{B}_{k-1}, diam(X_t)\} \geq diam(X_t)$, e no primeiro tempo t em que $\tilde{B}_k < diam(X_t)$ iniciamos uma nova fase. Portanto, temos que para todo $t \in fase_k$, $\|\ell_t\|_2 \leq 2\tilde{B}_k$.

Com isso podemos aplicar o desenvolvimento da desigualdade (4-5), e em particular a garantia do FTRL do Teorema 8, aos tempos da fase k para obter

$$\sum_{t\in fase_k} \langle \bar{x}_t - x_t, c_t - c_{true}\rangle \leq 8\eta_k T\tilde{B}_k^2 + \frac{D^2}{2\eta_k} = 4\tilde{B}_k D\sqrt{T},$$

onde na última desigualdade utilizamos a definição de η_k. Isso em todas as fases, temos

$$\sum_{t=1}^{T}\langle \bar{x}_t - x_t, c_t - c_{true}\rangle \leq 4D\sqrt{T}\sum_k \tilde{B}_k. \tag{4-6}$$

Para limitar esse somatório, temos a seguinte propriedade.

Lema 2 *Seja fim a última fase do algoritmo. Então $\sum_k \tilde{B}_k \leq 2\tilde{B}_{fim}$.*

Prova. Por definição temos $\tilde{B}_{k+1} \geq 2\tilde{B}_k$. Iterando essa propriedade, temos $\tilde{B}_{fim} \geq 2^{fim-k}\tilde{B}_k$. Portanto

$$\sum_{k=1}^{fim}\tilde{B}_k \leq \sum_{k=1}^{fim}\frac{\tilde{B}_{fim}}{2^{fim-k}} \leq \tilde{B}_{fim}\sum_{i=0}^{\infty}2^{-i} = 2\tilde{B}_{fim},$$

concluindo a prova do lema. ■

Por fim, note que $\tilde{B}_{fim} \leq 2\max_t diam(X_t) = 2B$, pois pela definição de $\tilde{B}_i$ nunca o setamos mais do que o dobro de um $diam(X_t)$ já visto. Aplicando isso e o Lema 2 à desigualdade 4-6 temos

$$\sum_{t=1}^{T}\langle \bar{x}_t - x_t, c_t - c_{true}\rangle \leq 8DB\sqrt{T}. \tag{4-7}$$

Isso conclui a prova do Teorema 9. ■

5
Aprender as Restrições

Neste capítulo consideramos o problema OInvOnline-LD, de se aprender o lado direito b_{true} do problema

$$\begin{aligned} \max \quad & c_t^T x \\ \text{s.t.} \quad & Ax \leq b_{true} \\ & x \geq 0, \end{aligned} \tag{5-1}$$

a partir do conhecimento de A e de observações dos c_t's e soluções ótimas x_t's (veja Seção (1.1) para relembrar a definição completa do problema).

Nossa segunda contribuição é o seguinte resultado.

Teorema 10 *Considere o problema* OInvOnline-LD. *Assuma que* $b_{true} \in [-B, B]^m$. *Então o algoritmo* AlgoLD *produz vetores* $\bar{b}^1, \ldots, \bar{b}^T \in [-B, B]^m$ *(estimativas para* b_{true}*) e* $\bar{x}_1, \ldots, \bar{x}_T$ *com as seguinte propriedades:*

1. *Para todo* t, $\bar{x}_t$ *é viável para* $\mathcal{X}$

2. *Essas soluções tem a seguinte garantia de valor:*

$$\frac{1}{T}\sum_{t=1}^{T}\langle c_t, x_t - \bar{x}_t\rangle \leq \frac{2n^{3/2}mCB\Delta}{T}, \tag{5-2}$$

onde $C = \max_t \|c_t\|_2$, *e* Δ *é o maior valor absoluto das matrizes* D^{-1}, *sendo* D *uma submatriz quadrada de* A *não singular.*

Novamente, como x_t é ótimo com relação a c_t, note que $\langle c_t, x_t - \bar{x}_t\rangle = |\langle c_t, x_t\rangle - \langle c_t, \bar{x}_t\rangle|$, e portanto essa garantia implica a garantia enunciada no Teorema 3.

A ideia do algoritmo é simples: para que as soluções $\bar{x}_t$ sejam sempre viáveis, temos que manter os estimadores $\bar{b}^t$ pequenos, ou seja, as restrições $Ax \leq \bar{b}^t$ mais apertadas do que as restrições originais $Ax \leq b_{true}$. Por isso, inicializamos $\bar{b}^t = (-B, \ldots, -B)$, o menor valor possível na faixa dada. Além disso, ao observarmos x_t, sua viabilidade traz uma cota inferior para b_{true}: $b_{true} \geq Ax_t$. Portanto, podemos setar nossa próxima estimativa como $\bar{b}^t = \max\{Ax_t, \bar{b}^{t-1}\}$ (o máximo é tomado a cada coordenada). Formalmente o algoritmo tem a seguinte descrição:

Algorithm 5 ALGOLD

Entrada: Matriz A, cota B tal que $b_{true} \in [-B, B]^m$

1: $\bar{b}^1 = (-B, \ldots, -B) \in \mathbb{R}^m$
2: **for** $t = 1$ to T **do**
3: Observa c^t
4: $\bar{x}_t \leftarrow \text{argmax}\{\langle c_t, x\rangle \mid Ax \leq \bar{b}^t\}$
5: Observa a solução ótima x_t ($\text{argmax}\{\langle c_t, x\rangle \mid Ax \leq b_{true}\}$)
6: $b^t \leftarrow Ax_t$
7: $\bar{b}^{t+1} \leftarrow \max\{b^t, \bar{b}^t\}$
8: **end for**

Como o algoritmo aumenta a estimativa $\bar{b}^t$ monotonamente e b_{true} está na região limitada $[-B, B]^m$, a intuição é que temos o "sanduiche" $\bar{b}^t \leq b_{true} \leq (B, \ldots, B)$ e portanto após um tempo deveríamos ter $\bar{b}^t \approx b_{true}$. Assim, podemos produzir soluções ótimas $\bar{x}_t$ viáveis, a partir das estimativas $(\bar{b}^1, \ldots, \bar{b}^T)$, que na média são essencialmente tão boas quanto as soluções ótimas observadas x_t.

5.1 Prova do Teorema 10

Item 1. Primeiro provamos que os vetores $\bar{x}_t$ computados pelo algoritmo satisfazem $A\bar{x}_t \leq b_{true}$. Para isso, primeiro demonstramos por indução que as estimativas $\bar{b}^t$ satisfazem $\bar{b}^t \leq b_{true}$. Por definição da cota B e $\bar{b}^1$, isso claramente vale para essa primeira iteração. Para o passo indutivo, pela viabilidade de x_t temos $b^t = Ax_t \leq b_{true}$, e portanto usando a hipótese indutiva $\bar{b}^{t+1} = \max\{b^t, \bar{b}^t\} \leq \max\{b_{true}, b_{true}\} = b_{true}$; isso conclui a indução.

Portanto, por definição de $\bar{x}_t$ temos $A\bar{x}_t \leq \bar{b}^t \leq b_{true}$, provando a viabilidade de $\bar{x}_t$.

Item 2. A seguinte é a principal observação para a análise de valor das soluções $\bar{x}_t$ computadas.

Lema 3 *Considere $\tilde{b}^t \in \mathbb{R}^m$ tal que $\tilde{b}^t \geq b^t$, e seja $\tilde{x}_t := \text{argmax}\{\langle c_t, \tilde{b}^t\rangle \mid Ax \leq \tilde{b}^t\}$. Então $\tilde{x}_t$ tem valor maior ou igual a x_t: $\langle c_t, \tilde{x}_t\rangle \geq \langle c_t, x_t\rangle$.*

Prova. Como por definição $b^t = Ax_t$, temos

$$\langle c_t, \bar{x}_t\rangle = \text{argmax}\{\langle c_t, x\rangle \mid Ax \leq \bar{b}^t\} \geq \text{argmax}\{\langle c_t, x\rangle \mid Ax \leq b^t\} \geq \langle c_t, x_t\rangle,$$

onde a primeira desigualdade vem do fato de que um conjunto viável está contido no outro. ■

Se pudéssemos garantir $\bar{b}^t \geq b^t$ para todo t, teríamos então a comparação desejada $\langle c_t, \bar{x}_t \rangle \geq \langle c_t, x_t \rangle$. Como esse não é o caso, introduzimos o majorante $\hat{b}^t = \max\{\bar{b}^t, b^t\}$ (então por definição $\hat{b}^t \geq b^t$). A ideia é provar que "na media" temos $\bar{b}^t \gtrsim \hat{b}^t$, e portanto podemos aplicar de forma aproximada o lema acima para obter $\langle c_t, \bar{x}_t \rangle \gtrsim \langle c_t, x_t \rangle$. O seguinte lema formaliza o primeiro passo.

Lema 4 *Temos* $\sum_{t=1}^{T} \|\bar{b}^t - \hat{b}^t\|_1 \leq 2mB$.

Prova. Note que por definição de $\hat{b}^t$ temos $\hat{b}^t \geq \bar{b}^t$, e portanto

$$\|\bar{b}^t - \hat{b}^t\|_1 = \sum_{i=1}^{m} \left|\bar{b}_i^t - \hat{b}_i^t\right| = \sum_{i=1}^{m} \left(\hat{b}_i^t - \bar{b}_i^t\right).$$

Além disso, por definição de atualização das estimativas $\bar{b}^t$ temos $\bar{b}^{t+1} = \hat{b}^t$. Combinando com a observação anterior, temos

$$\sum_{t=1}^{T} \|\bar{b}^t - \hat{b}^t\|_1 = \sum_{t=1}^{T} \sum_{i=1}^{m} \left(\bar{b}_i^{t+1} - \bar{b}_i^t\right) = \sum_{i=1}^{m} \left(\bar{b}_i^{T+1} - \bar{b}_i^1\right) \leq 2mB,$$

onde a última desigualdade segue do fato que $\bar{b}^t \in [-B, B]^m$. Isso conclui a prova. ■

Agora defina $\hat{x}_t := \text{argmax}\{\langle c_t, x \rangle \mid Ax \leq \hat{b}^t\}$, que pelo Lema 3 satisfaz

$$\langle c_t, \hat{x}_t \rangle \geq \langle c_t, x_t \rangle \tag{5-3}$$

Note que $\hat{x}_t$ e $\bar{x}_t$ são soluções ótimas do mesmo problema de otimização com dois lados direito diferentes $\hat{b}_t$ e $\bar{b}^t$, respectivamente. Pra quantificar a distância entre essa soluções ótimas em função da diferença entre os lados direito, utilizamos o seguinte resultado clássico.

Teorema 11 (Teorema 10.5 de (42)) *Considere dois programas lineares que diferem apenas no lado direto das restrições:* $\max\{\langle f, x \rangle \mid Mx \leq d'\}$ *e* $\max\{\langle f, x \rangle \mid Mx \leq d''\}$, *onde* $M \in \mathbb{R}^{n \times m}$, $f \in \mathbb{R}^n$, *e* $d', d'' \in \mathbb{R}^m$. *Sendo* x' *e* x'' *soluções dos problemas, respectivamente, temos*

$$\|x' - x''\|_\infty \leq n\Gamma \|d' - d''\|_\infty, \tag{5-4}$$

onde Γ *é o maior valor absoluto das matrizes* D^{-1}, *sendo* D *uma submatriz quadrada de* M *não singular.*

Como definimos Δ precisamente como o quantidade Γ do lema anterior mas com relação à matriz A, temos

$$\|\bar{x}_t - \hat{x}_t\|_\infty \leq n\Delta \|\bar{b}_t - \hat{b}_t\|_\infty \leq n\Delta \|\bar{b}_t - \hat{b}_t\|_1. \tag{5-5}$$

Para terminar o argumento, precisamos compara o lado direito dessa expressão com a diferença de valor entre $\bar{x}_t$ e $\hat{x}_t$.

Lema 5 *Lembrando que* $C = \max_t \|c_t\|_2$, *temos* $\langle c_t, \hat{x}_t - \bar{x}_t \rangle \leq \sqrt{n} C \|\bar{x}_t - \hat{x}_t\|_\infty$.

Prova. Aplicando a desigualdade Cauchy-Schwarz, temos

$$\langle c_t, \hat{x}_t - \bar{x}_t \rangle \leq \|c_t\|_2 \|\bar{x}_t - \hat{x}_t\|_2 \leq C \|\bar{x}_t - \hat{x}_t\|_2.$$

Porém note que $\|\bar{x}_t - \hat{x}_t\|_2 \leq \sqrt{n} \|\bar{x}_t - \hat{x}_t\|_\infty$ (pois a distancia ℓ_2 é maximizada quando todas as coordenadas dos vetores diferem de $\|\bar{x}_t - \hat{x}_t\|_\infty$ unidades. Isso conclui a prova. ■

Combinando o Lema 5, as desigualdades (5-3) e (5-5), e o Lema 4, temos que

$$\sum_{t=1}^{T} \langle c_t, x_t - \bar{x}_t \rangle \leq 2n^{3/2} mCB\Delta.$$

Dividindo por T obtemos a prova do item 2 do Teorema 10, concluindo assim a sua prova.

6
Experimentos computacionais

Nossa terceira contribuição são experimentos computacionais dos algoritmos desenvolvidos. Como não existem instâncias padrão de problemas de otimização inversa online, consideramos adaptações de problemas clássicos de otimização combinatória a esse contexto. Para o modelo OInvOnline-FO consideramos os problemas da mochila, de caminho mais curto com restrição de recurso, multifluxo em rede e planejamento de produção; para o modelo OInvOnline-LD consideramos o problema da mochila multidimensional.

Nossos resultados computacionais foram obtidos por meio de um Notebook Samsung (2016), com Intel core i5 com 2.2GHz cores, implementamos usando Julia através do JuMP e CPLEX Solver.

6.1
Aprender Preferências dos Clientes

Considere um mercado que oferece G produtos, onde os preços desses produtos variam a cada dia $t \in T$. Assumimos que os produtos são escolhidos para serem comprados pelos seus clientes de forma a maximizar a utilidade, dado suas restrições de gastos. No dia t, o cliente resolve o seguinte problema de otimização,

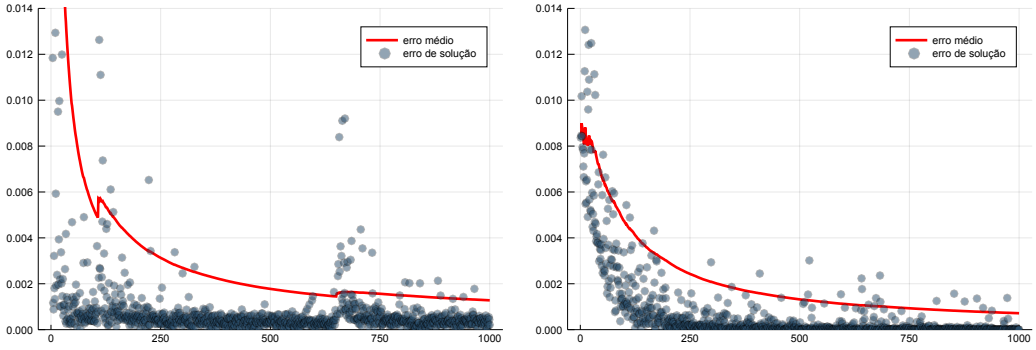
$$\begin{aligned} \max \quad & \sum_{g \in G} c_{true,g} x_g \\ \text{s.t.} \quad & \sum_{g \in G} p_g^t x_g \leq p_0^t \\ & x \in \{0,1\}^{|G|}, \end{aligned}$$

onde c_{true} é a função utilidade desconhecida que buscamos aprender, p_g^t é preço do produto g e p_0^t representa o máximo que o cliente pode gastar na compra dos produtos.

Nós geramos instâncias para esse problema baseado no data set para o problema da mochila *knapPI_2_500_1000_1* usado em (43). Esse dataset contém 1 instância do problema da mochila com 500 itens e capacidade máxima da mochila igual a 1000. O vetor de utilidade c_{true} desconhecido corresponde ao valor dos itens da mochila das instâncias normalizado para obter um vetor c_{true} como uma distribuição de probabilidade. O preço do item g no dia t assumiu

o valor $p_g^t = w + r_g^t$, no qual w é o peso associado a cada item da mochila das instâncias, e r_g^t é escolhido aleatoriamente das instâncias no intervalo $[-10, 10]$, por meio de uma distribuição uniforme. Por fim, o lado direito p_0^t, restrição de gastos do cliente no dia t, assumiu um valor aleatório dado por uma distribuição uniforme no intervalo $[W_{min}, W_{max}]$, onde W_{min} é a média do peso dos itens no dia t e W_{max} é a capacidade máxima original da mochila, que nessas instâncias é 1000. Setamos o horizonte de tempo $T = 1000$.

Munidos dessas instâncias do problema de otimização inversa online, realizamos experimentos computacionais com o algoritmo OOFL (algoritmo que usa o MWU de (32)) e o nosso algoritmo AlgoFO.



(a) AlgoFO (b) OOFL

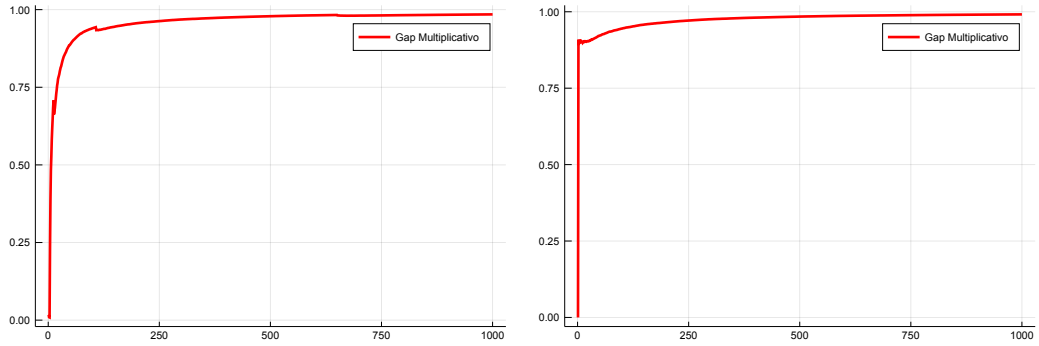
Figura 6.1: Em (a) plotamos o erro de solução em cada tempo t e o erro de solução médio do algoritmo AlgoFO, assim como em (b) para o algoritmo OOFL

Consideramos o erro de solução em cada instante de tempo $t \in T$, dado por $|\langle c_{true}, \bar{x}_t\rangle - \langle c_{true}, x_t\rangle|$, o qual descreve a diferença entre a solução ótima $\bar{x}_t$, obtida pela resolução do problema com a estimativa da função objetivo, i.e. c_t, e a solução ótima $x_t \in X_t$ obtida otimizando o problema original, ou seja, com a função objetivo desconhecida c_{true}. E também o erro de solução médio, definido como $\frac{1}{t}\sum_{t'=1}^{t} |\langle c_{true}, \bar{x}_{t'}\rangle - \langle c_{true}, x_{t'}\rangle|$, que é dado pela média do erro de solução em cada instante de tempo $t \in T$. A Figura (6.1) apresenta o erro de solução em cada instante t e o erro médio para os algoritmos OOFL e AlgoFO. Observe que para esta instância o erro de solução em cada instante de tempo do algoritmo AlgoFO converge mais rapidamente que o algoritmo OOFL enquanto que o erro médio se mostra equivalente para os dois algoritmos.

Agora, para avaliar a convergência entre a soluções na função objetivo consideramos o gap multiplicativo entre as soluções ótimas x_t e as soluções $\bar{x}_t$. Mais precisamente plotamos,

$$\frac{\sum_{t'=1}^{t} c_{true}\bar{x}_{t'}}{\sum_{t'=1}^{t} c_{true}x_{t'}},$$

como demonstra a Figura (6.2), o AlgoFO tem uma performance um pouco superior a do algoritmo OOFL.



(a) AlgoFO (b) OOFL

Figura 6.2: Plotamos o gap multiplicativo entre as soluções $\bar{x}_t$ e x_t na função objetivo c_{true}, dada por $\frac{\sum_{t'=1}^{t} c_{true}\bar{x}_{t'}}{\sum_{t'=1}^{t} c_{true}x_{t'}}$ em (a) para o AlgoFO, assim como em (b) para o OOFL

6.2 Aprender Tempos de Viagem

Neste exemplo, consideramos uma rede rodoviária, onde observamos um motorista em cada tempo $t \in T$. Mais precisamente consideramos, o problema do caminho mais curto com restrição de recurso, no qual dado um grafo G(V, E) direcionado, os motoristas devem encontrar o caminho de s para t de custo mínimo sujeito a restrições de recurso. Onde a_e representa o comprimento de cada aresta $e \in E$.

As observações $t \in [T]$ representam motoristas em uma rede, e cada observação consiste em $p_t = (p_t^1, p_t^2, p_t^3)$, sendo p_t^1 o nó de origem e p_t^2 o nó de destino da viagem do motorista t. Assumimos que cada motorista pega o caminho com o menor tempo de viagem em relação aos tempos de viagem desconhecido c_e com $e \in E$ estando sujeito ao limite da distancia total que pode ser percorrida p_t^3. Os valores de x_t indicam a aresta do grafo que o motorista

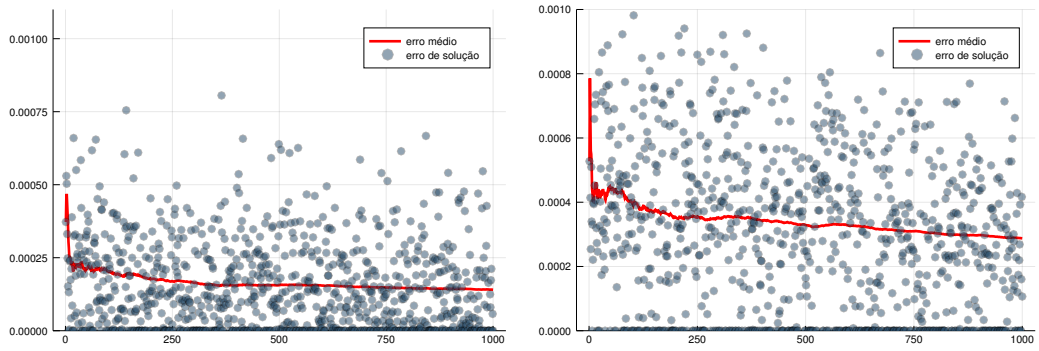
t atravessou. O motorista t resolve o seguinte problema de otimização:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_{true,e} x_e \\
\text{s.t.} \quad & \sum_{e \in \delta^-(v)} x_e - \sum_{e \in \delta^+(v)} x_e = \begin{cases} -1, & \text{se } v = p_t^1 \\ 1, & \text{se } v = p_t^2 \\ 0, & \text{caso contrário} \end{cases} \quad (\forall v \in V) \\
& \sum_{e \in E} a_e x_e \leq p_t^3 \\
& x \in \{0,1\}^{|E|},
\end{aligned}$$

e queremos aprender os valore de $c_{true,e}$ que correspondem ao tempo de viagem de atravessar o arco e.

Para este problema geramos instâncias baseado no *dataset* para o problema do caminho mais curto com restrição de recurso $rcsp18$ da OR-LIBRARY, contendo 500 vértices e 4858 arestas. O valor do vetor desconhecido de tempos de viagem $c_{true,e}$ corresponde ao custo das arestas das instâncias normalizado para obter uma distribuição de probabilidade. O valor do comprimento a_e de cada aresta corresponde ao valor k que representa a quantidade de recurso consumida ao se passar pelo vértice $i \in V$ das instâncias. Em cada tempo t, escolhemos um par aleatório de nós de origem e destino e assumimos que o valor de p_t^3, que limita a distancia total percorrida, é descrito por um número aleatório através de uma distribuição uniforme no intervalo $[r_{min}, r_{max}]$ onde r_{min} e r_{max} são, respectivamente, os valores de recurso mínimo e o máximo das instâncias. Setamos o horizonte de tempo $T = 1000$.

Com as instâncias do problema de otimização inversa online, realizamos experimentos computacionais aplicando os algoritmos OOFL e o AlgoFO, para comparar seus desempenhos. Aqui também consideramos o erro de solução médio e em cada instante $t \in T$.



(a) AlgoFO (b) OOFL

Figura 6.3: Erro e o erro médio em cada instante t para o problema de aprender os tempo de viagem.

Podemos observar na Figura (6.3) que tanto para o erro da solução no instante $t \in T$ quanto para o erro médio, o AlgoFO obtém melhores resultados que o OOFL, para essa instância do problema, além de uma melhor convergência. É possível também identificar, que o erro da solução em relação ao resultado apresentado pela Figura (6.1) tem uma convergência mais lenta, o que pode ser explicado pela maior complexidade do problema.

6.3 Aprender Custos de Transporte

Consideramos uma rede de transporte que liga fábricas a seus clientes e assumimos que ao todo K mercadorias que são produzidas nas fábricas. De maneira geral, consideramos aqui o problema de multifluxo em rede, onde a rede pode ser descrita como um grafo direcionado $G(N, A)$ e cada fábrica é um nó $n \in N$ de origem e cada consumidor que demanda a mercadoria k é um nó $n \in N$ de destino. Assim, em cada dia $t \in [T]$ o proprietário das fábricas observa as demandas e capacidades de transporte para transportar as mercadorias demandadas, através da rede, a custo mínimo. Para cada arco $a \in A$ e cada mercadoria $k \in K$ temos um custo de transporte $c^k_{true,a}$ desconhecido associado.

As observações $t \in [T]$ são dadas por $p_t = (p^t_1, p^{t,k}_2)$ onde p^t_1 representa a capacidade total de transporte em cada arco $a \in A$, que podemos associar as capacidades dos veículos de transporte que circulam na rede, e $p^{t,k,n}_2$ é a demanda do item k para cada cliente $n \in N$. O proprietário deve atribuir os custos $c^k_{true,a}$ de transportar cada mercadoria através dos arcos $a \in A$ da rede, dada as observações p_t e quantidade da mercadoria k que passa pelo arco $a \in A$ no dia t. Assim, a cada dia $t \in [T]$ o proprietário deseja resolver o problema

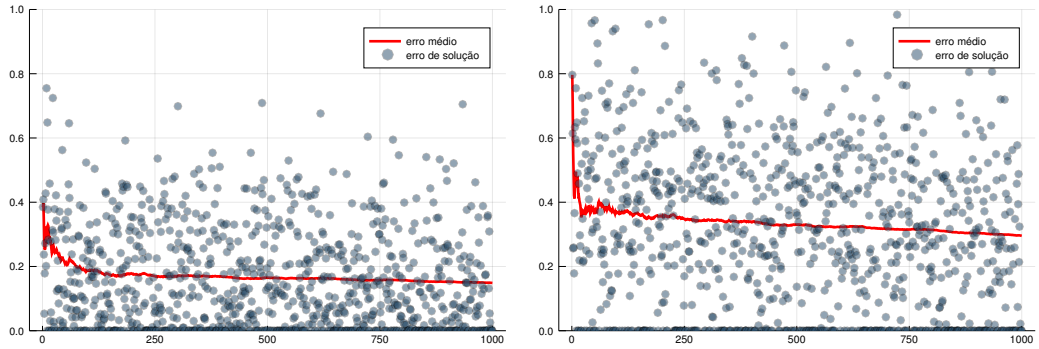
de multifluxo em rede dado por

$$\begin{aligned} \min \quad & \sum_{k\in K}\sum_{a\in A} c^k_{true,a} x^k_a \\ \text{s.t.} \quad & \sum_{k\in K} x^k_a \leq p^t_1 \quad \forall a \in A \\ & \sum_{a\in \delta_-} x^k_e - \sum_{a\in \delta_+} x^k_a = p^{t,k,n}_2 \quad \forall n \in N, \quad \forall k \in K \\ & x \geq 0, \end{aligned}$$

onde x^k_a é a quantidade da mercadoria k que passando pela arco $a \in A$.

Construímos instâncias baseadas no *dataset* $AerTrans$, disponível em `http://www.di.unipi.it/optimize/Data/MMCF.html`, com $K = 40$ mercadorias, $V = 49$ vértices e $E = 137$ arestas. O valor do vetor desconhecido de custos de transporte $c^k_{true,a}$ foi associado ao valor do custos das arestas das instâncias normalizado para $c^k_{true,a}$ ser uma distribuição de probabilidade. No dia t o valor da demanda, para os nós fábricas, $p^{k,t,n}_2 = q^f_k + s_t$, onde q^f_k é a quantidade da mercadoria k produzida pela fábrica f das instâncias, e s^t_f é escolhido aleatoriamente com números no intervalo $[-3, 3]$. Agora para os nós clientes associamos a demanda $p^{k,t,n}_2$ a soma da quantidade produzida da mercadoria k pelas fábricas multiplicado por -1 (por ser um nó de demandante), caso o nó não seja fábrica nem cliente a demanda $p^{k,t,n}_2 = 0$. O valor das capacidades das arestas corresponde a um número aleatório dado por uma distribuição uniforme no intervalo $[v_{min}, v]$, onde v_{min} é o menor valor que a capacidade da aresta pode assumir e v é o valor de capacidade da aresta das instâncias.

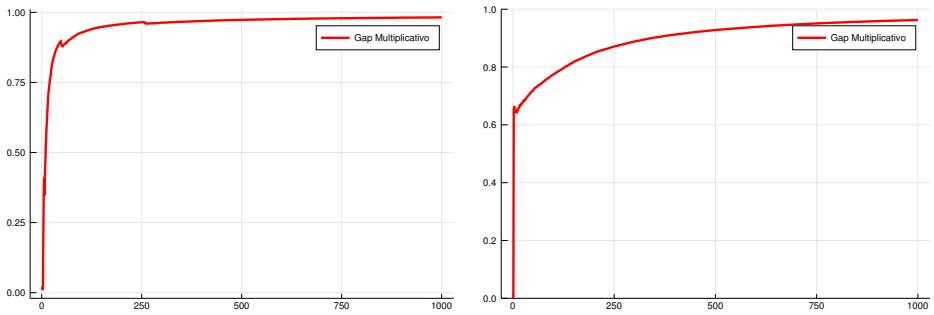
Munidos das instâncias para o problema de otimização online, realizamos experimentos computacionais com os algoritmos OOFL e AlgoFO. Assim, computamos o erro de solução no tempo t e o erro de solução médio, bem como avaliamos a convergência das soluções na função objetivo c_{true}.



(a) AlgoFO (b) OOFL

Figura 6.4: Erro de solução em cada instante $tv \in T$ e o erro médio para os algoritmos AlgoFO, em (a) e OOFL, em (b).

Observe que na Figura (6.4) mais uma vez o AlgoFO converge mais rapidamente que o OOFL, além de apresentar taxas de erro menores. Assim, também avaliamos a convergência das soluções produzidas pelo algoritmo e a solução ótima na função objetivo c_{true} por meio do Gap multiplicativo, Figura (6.5). As soluções $\bar{x}_t$ produzidas pelo AlgoFO convergem mais rapidamente na função objetivo c_{true} em relação as soluções ótimas conhecidas do que as soluções obtidas pelo algoritmo OOFL.



(a) AlgoFO (b) OOFL

Figura 6.5: Gap multiplicativo entre as soluções $\bar{x}_t$ e x_t na função objetivo c_{true} para o AlgoFO (a) e para o OOFL (b).

6.4 Aprender Custos de Produção

Agora, consideramos uma fábrica que recebe, a cada mês $t \in T$, N demandas de seus clientes. Mais precisamente, consideramos o problema de planejamento de produção, onde observamos em cada mês as demandas dos clientes que devem ser atendidas minimizando os custos totais da produção.

No mês t observamos $p_t = (p_1^t, p_2^t, p_3^t, p_4^t)$, no qual p_1^t são os tempos de produção de cada produto $i = \{1, ..., N\}$, p_2^t é a capacidade ou tempo máximo

de produção, p_3^t representa a quantidade em estoque dos produtos e p_4^t são as demanda dos clientes. A fábrica deve atribuir o custo c_i de produzir o produto i, o custo h_i de manter determinado produto i no estoque e o custo s_i de configuração da produção dado as observações p_t bem como a quantidade de produtos produzidos e em estoque, além de saber que produtos são produzidos em uma dada configuração de produção no mês t. De modo geral,a cada mês t a fábrica quer resolver seguinte problema de planejamento de produção ,

$$\begin{aligned}
\min \quad & \sum_{i \in N} c_i x_i + s_i y_i + h_i z_i \\
\text{s.t.} \quad & \sum_{i \in N} p_1^t x_i \leq p_2^t \\
& p_1^t x_i \leq p_2^t y_i \quad \forall i \in N \\
& p_3^t + x_i + z_i = p_4^t \quad \forall i \in N \\
& x, z \geq 0 \\
& y \in \{0,1\}^{|N|},
\end{aligned}$$

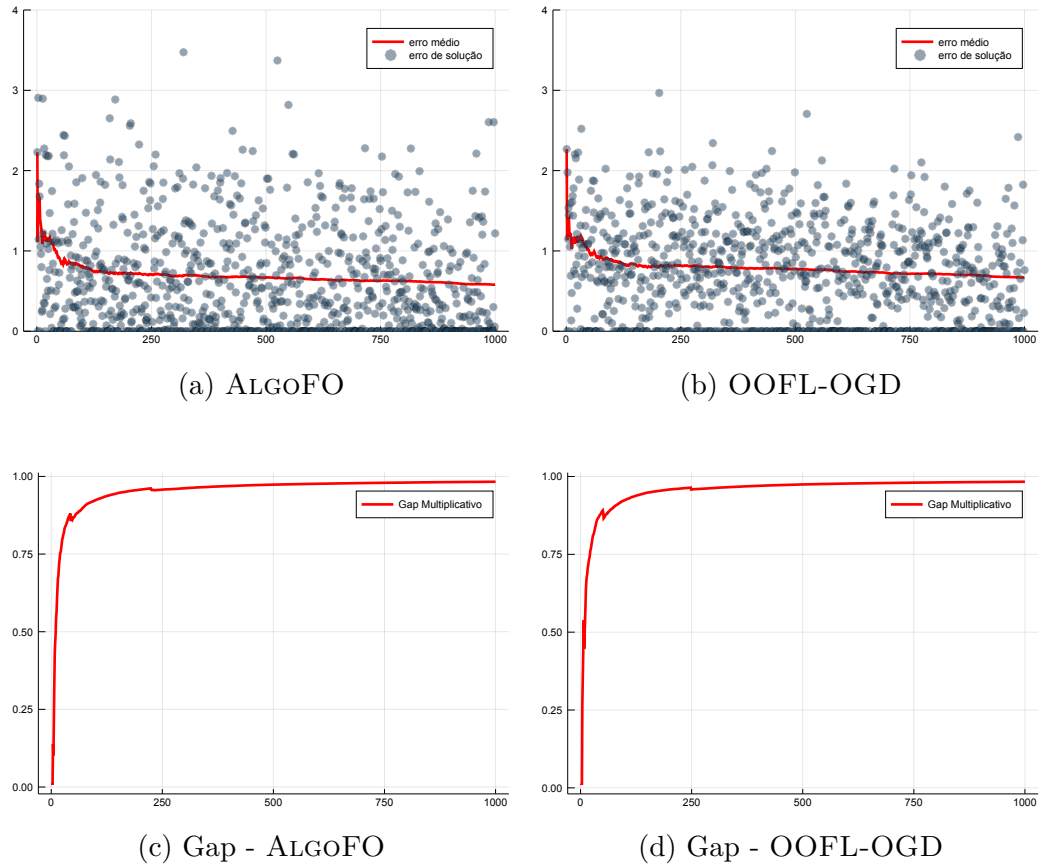
onde x_i é a quantidade produzida do item i, y_i denota se o item i foi produzido ou não e z_i representa o inventário ou estoque de cada produto i, buscando aprender os custos, c_i, s_i e h_i para melhor gerenciamento de seus recursos e aprimorar planejamentos de produção futuros.

Criamos instâncias para o problema, baseadas no *dataset SetA* da CSPLib, especifico para o *Lot Sizing Problem* com $N = 20$ produtos. O valor atribuído aos vetores desconhecidos c_i, s_i e h_i são, respectivamente, os custos de produção, de configuração e de estoque dos produtos das instâncias. O *dataset* conta com uma matriz para cada parâmetro do problema, dado um horizonte de tempo de 16 dias.

Por ser um horizonte de tempo pequeno, setamos $T = 1000$ e para cada instante t o tempo de produção p_1^t de cada produto i corresponde a um número aleatório descrito de uma distribuição no intervalo $[k_{min}, k_{max}]^N$, onde k_{min} e k_{max} são, respectivamente, o menor e o maior elemento da matriz de tempos de produção para o produto i, associamos o valor da demanda p_4^t de maneira análoga. Já o valor do tempo máximo de produção p_2^t é definido por uma distribuição uniforme no intervalo entre o menor tempo limite de produção e a média dos tempos limites. E por fim, a quantidade de produtos em estoque assume o valor $p_3^t = e + r_t$, no qual e é a média da quantidade de itens em estoque para o produto i e r_t é um número aleatório distribuído uniformemente no entre $[-10, 10]$.

Com as instâncias para o problema de otimização inversa online realizamos experimentos computacionais com os algoritmos AlgoFO e OOFL-

OGD (algoritmo de Barmann et.al (33) que generaliza c_{true}) para aprender os custos associados a produção c_i, s_i e h_i desconhecidos. Computamos os erros de solução por instante t e médio, bem como a razão do custo médio, Figura (6.6) (a) e (b). Além disso, computamos também a convergência entre as soluções $\bar{x}_t$ e x_t na função objetivo real (ver Figura (6.6) (c) e (d)).



(a) AlgoFO (b) OOFL-OGD

(c) Gap - AlgoFO (d) Gap - OOFL-OGD

Figura 6.6: Erro por iteração t e o erro médio para o algoritmo AlgoFO em (a) e para o algoritmo OOFL-OGD em (b). Em (c) plotamos o gap multiplicativo entre as soluções $\bar{x}_t$ e x_t para o algoritmo AlgoFO e em (d) para o algoritmo OOFL-OGD.

Observe que os dois algoritmos tem um resultado bastante similar, tanto para os erro de solução no tempo t e médio que convergem em uma faixa de valor bem próxima, quanto em relação aos gaps multiplicativos entre as soluções produzidas pelos algoritmos e as soluções ótimas.

6.5 Aprender Capacidades de Recurso

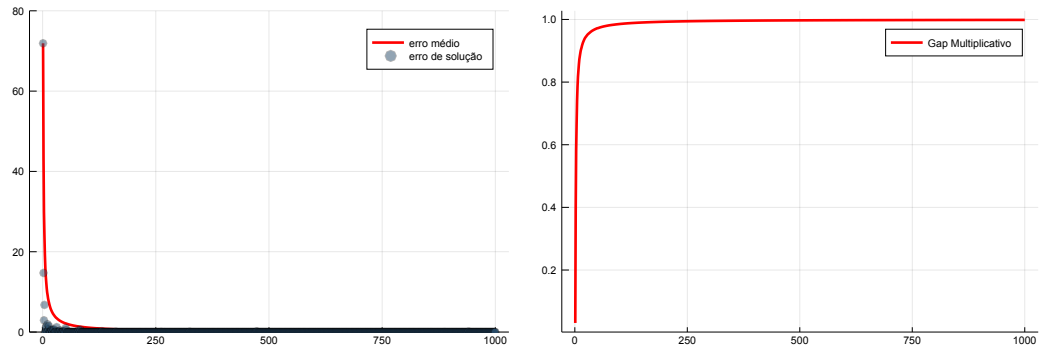
Para testar o funcionamento do Algoritmo AlgoLD para o problema OInvOnline-LD, consideramos o problema da mochila multidimensional. O problema considera um conjunto de W diferentes itens, cada qual com um valor

associado e S diferentes recursos cada um com sua capacidade. Consideramos a versão online do problema, no qual assumimos ter um horizonte de tempo $t = 1, ..., T$ e que conhecemos a quantidade $r_{s,w}$ do recurso $s \in S$ que cada item $w \in W$ consome. Assumimos também que os itens são escolhidos de forma a maximizar o valor da mochila sem extrapolar a capacidade de nenhum dos recursos. No instante t, observamos o valor v_w do item w e queremos resolver o seguinte problema de otimização,

$$\begin{aligned} \max \quad & \sum_{w \in W} v_w^t x_w \\ \text{s.t.} \quad & \sum_{w \in W} r_{s,w} x_w \leq b_{true,s} \quad \forall s \in S \\ & x \in \{0,1\}^{|W|}, \end{aligned}$$

onde x_w indica se o item w está ou não na mochila e b_s é a capacidade de recurso desconhecida que buscamos aprender.

Construímos instâncias para o problema, baseadas no data set *mknapcb*4 da OR-LIBRARY, contendo $W = 500$ itens, $S = 30$ recursos e T = 1000. Em cada tempo t, o valor dos itens v_w^t é dado por uma distribuição uniforme no intervalo $[\min\{p_w\}, \max\{p_w\}]$, onde p_w é o valor do item no data set. As capacidades de recurso r_w são associadas ao seu respectivo valor das instâncias.



(a) Erro por rodada (b) Gap entre as soluções $\bar{x}_t$ e x_t

Figura 6.7: Em (a) plotamos o erro em cada tempo t e o erro médio do algoritmo , assim como em (b) o gap multiplicativo entre as soluções $\bar{x}_t$ e x_t, dada por $\frac{\sum_{t'=1}^{t} c_{t'} \bar{x}_{t'}}{\sum_{t'=1}^{t} c_{t'} x_{t'}}$.

O erro da solução no tempo t é definido por $|\langle c_t, \bar{x}_t \rangle - \langle c_t, x_t \rangle|$, o qual descreve a diferença entre a solução $\bar{x}_t$, obtida pela resolução do problema com o lado direito das restrições definido por $\bar{b}_t$, e x_t, solução obtida com b_{true}, na função objetivo c_t. O erro médio da solução é dado por $\frac{1}{t} \sum_{t'=1}^{t} |\langle c_{t'}, \bar{x}_{t'} \rangle - \langle c_{t'}, x_{t'} \rangle|$. A Figura (6.7) mostra que esse erro vai se tornando pequeno a medida que t cresce, além de mostrar que $\bar{x}_t$ é tão bom quanto x_t na função objetivo.

7
Conclusões

Este capítulo apresenta as observações finais sobre nosso trabalho. Nós discutimos nossas contribuições e apresentamos as direções para trabalhos futuros.

Nesta dissertação demonstramos como aprender a função objetivo e as restrições de problema de otimização online através da observação da sua solução ótima no decorrer de múltiplas rodadas. Nossa abordagem faz uso de técnicas de *Online Learning* por meio da qual desenvolvemos dois algoritmos, AlgoFO que aprende a função objetivo e AlgoLD que aprende as restrições nos permitindo produzir soluções tão boas quanto as ótima no decorrer em poucas observações.

Como mencionado na Introdução, na nossa primeira contribuição consideramos o problema OInvOnline-FO e generalizamos os resultados de Barmann et.al (32) para um vetor c_{true} qualquer, removendo qualquer hipótese de que ele deveria ser uma distribuição de probabilidade através do algoritmo AlgoFO, descrito no Capítulo (4), com garantia dada pelo Teorema (9). Mesmo obtendo o mesmo resultado que Barmann et.al em (33) removemos a hipótese adicional que o diâmetro máximo dos conjuntos viáveis do problema X_t são conhecidos a priori.

Nossa segunda contribuição foi apresentar uma solução para o problema OInvOnline-LD, que foi deixado em aberto por Barmann et.al em (32, 33), através do algoritmo AlgoLD, descrito no Capítulo (5), com garantia dada pelo Teorema (10). Nossa última contribuição foram experimentos computacionais que demostraram a aplicabilidade da nossa abordagem algorítmica e sua eficácia. Investigamos seus usos em problemas de otimização combinatória clássicos e obtivemos resultados similares aos algoritmos existentes.

Como próximos passos poderíamos estender o problema de otimização inversa online para aprender em conjunto a função objetivo e o lado direito das restrições de problema de otimização online e obter uma garantia satisfatória para seu uso em aplicações reais.

Referências bibliográficas

[1] IYENGAR, G.; KANG, W.. **Inverse conic programming with applications**. Operations Research Letters, 33(3):319–330, 2005.

[2] XU, S. J.; NOURINEJAD, M.; LAI, X. ; CHOW, J. Y.. **Network learning via multiagent inverse transportation problems**. Transportation Science, 52(6):1347–1364, 2018.

[3] CHOW, J. Y.; RECKER, W. W.. **Inverse optimization with endogenous arrival time constraints to calibrate the household activity pattern problem**. Transportation Research Part B: Methodological, 46(3):463–479, 2012.

[4] LEE, T.. **Generalized Inverse Optimization with Application to Cancer Therapy**. PhD thesis, University of Toronto, 2015.

[5] CHAN, T. C.; LEE, T. ; TEREKHOV, D.. **Inverse optimization: Closed-form solutions, geometry, and goodness of fit**. Management Science, 2018.

[6] BIRGE, J. R.; HORTAÇSU, A. ; PAVLIN, J. M.. **Inverse optimization for the recovery of market structure from market outcomes: An application to the miso electricity market**. Operations Research, 65(4):837–855, 2017.

[7] TROUTT, M. D.; PANG, W.-K. ; HOU, S.-H.. **Behavioral estimation of mathematical programming objective function coefficients**. Management science, 52(3):422–434, 2006.

[8] BEIL, D. R.; WEIN, L. M.. **An inverse-optimization-based auction mechanism to support a multiattribute rfq process**. Management Science, 49(11):1529–1545, 2003.

[9] BURTON, D.; TOINT, P. L.. **On an instance of the inverse shortest paths problem**. Mathematical Programming, 53(1-3):45–61, 1992.

[10] BURTON, D.; TOINT, P. L.. **On the use of an inverse shortest paths algorithm for recovering linearly correlated costs**. Mathematical Programming, 63(1-3):1–22, 1994.

[11] ZHANG, J.; MA, Z. ; YANG, C.. **A column generation method for inverse shortest path problems**. Zeitschrift für Operations Research, 41(3):347–358, 1995.

[12] ZHANG, B.; GUAN, X.; PARDALOS, P. M. ; HE, C.. **An algorithm for solving the shortest path improvement problem on rooted trees under unit hamming distance**. Journal of Optimization Theory and Applications, 178(2):538–559, 2018.

[13] AHUJA, R. K.; ORLIN, J. B.. **Inverse optimization**. Operations Research, 49(5):771–783, 2001.

[14] GALLEGO, J. S.; MADSEN, H.. **Inverse Optimization and Forecasting Techniques Applied to Decision-making in Electricity Markets**. PhD thesis, Technical University of Denmark (DTU), 2017.

[15] TROUTT, M. D.; BRANDYBERRY, A. A.; SOHN, C. ; TADISINA, S. K.. **Linear programming system identification: The general nonnegative parameters case**. European Journal of Operational Research, 185(1):63–75, 2008.

[16] AHUJA, R. K.; ORLIN, J. B.. **A faster algorithm for the inverse spanning tree problem**. Journal of Algorithms, 34(1):177–193, 2000.

[17] ZHANG, J.; LIU, Z.. **Calculating some inverse linear programming problems**. Journal of Computational and Applied Mathematics, 72(2):261–273, 1996.

[18] HEUBERGER, C.. **Inverse combinatorial optimization: A survey on problems, methods, and results**. Journal of combinatorial optimization, 8(3):329–361, 2004.

[19] HE, Y.; ZHANG, B. ; YAO, E.. **Weighted inverse minimum spanning tree problems under hamming distance**. Journal of Combinatorial Optimization, 9(1):91–100, 2005.

[20] AMAN, M.; HASSANPOUR, H. ; TAYYEBI, J.. **A modified algorithm for solving the inverse minimum cost flow problem under the bottleneck-type hamming distance**. Bulletin of the Transilvania University of Brasov. Mathematics, Informatics, Physics. Series III, 9(1):97, 2016.

[21] ZHANG, X.; WANG, Q. ; ZHOU, J.. **Two uncertain programming models for inverse minimum spanning tree problem**. Industrial Engineering and Management Systems, 12(1):9–15, 2013.

[22] ZHANG, B.; ZHANG, J. ; HE, Y.. **Constrained inverse minimum spanning tree problems under the bottleneck-type hamming distance**. Journal of Global Optimization, 34(3):467–474, 2006.

[23] HOCHBAUM, D. S.. **Efficient algorithms for the inverse spanning-tree problem**. Operations Research, 51(5):785–797, 2003.

[24] CHATALBASHEV, V.. **Inverse convex optimization**.

[25] KESHAVARZ, A.; WANG, Y. ; BOYD, S.. **Imputing a convex objective function**. In: 2011 IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL, p. 613–619. IEEE, 2011.

[26] ASWANI, A.; SHEN, Z.-J. ; SIDDIQ, A.. **Inverse optimization with noisy data**. Operations Research, 66(3):870–892, 2018.

[27] HUANG, S.. **Inverse problems of some np-complete problems**. In: INTERNATIONAL CONFERENCE ON ALGORITHMIC APPLICATIONS IN MANAGEMENT, p. 422–426. Springer, 2005.

[28] SCHAEFER, A. J.. **Inverse integer programming**. Optimization Letters, 3(4):483–489, 2009.

[29] WANG, L.. **Cutting plane algorithms for the inverse mixed integer linear programming problem**. Operations Research Letters, 37(2):114–116, 2009.

[30] DUAN, Z.; WANG, L.. **Heuristic algorithms for the inverse mixed integer linear programming problem**. Journal of Global Optimization, 51(3):463–471, 2011.

[31] ERKIN, Z.; BAILEY, M. D.; MAILLART, L. M.; SCHAEFER, A. J. ; ROBERTS, M. S.. **Eliciting patientsŕevealed preferences: an inverse markov decision process approach**. Decision Analysis, 7(4):358–365, 2010.

[32] BÄRMANN, A.; POKUTTA, S. ; SCHNEIDER, O.. **Emulating the expert: Inverse optimization through online learning**. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 400–410, 2017.

[33] BÄRMANN, A.; MARTIN, A.; POKUTTA, S. ; SCHNEIDER, O.. **An online-learning approach to inverse optimization**. arXiv preprint arXiv:1810.12997, 2018.

[34] DONG, C.; CHEN, Y. ; ZENG, B.. **Generalized inverse optimization through online learning**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 86–95, 2018.

[35] DUIN, C.; VOLGENANT, A.. **Some inverse optimization problems under the hamming distance**. European Journal of operational research, 170(3):887–899, 2006.

[36] SOKKALINGAM, P.; AHUJA, R. K. ; ORLIN, J. B.. **Solving inverse spanning tree problems through network flow techniques**. Operations Research, 47(2):291–298, 1999.

[37] ROLAND, J.; FIGUEIRA, J. R. ; DE SMET, Y.. **The inverse $\{0, 1\}$-knapsack problem: theory, algorithms and computational experiments**. Discrete Optimization, 10(2):181–192, 2013.

[38] SHALEV-SHWARTZ, S.. **Online learning and online convex optimization**. Foundations and Trends® in Machine Learning, 4(2):107–194, 2012.

[39] HAZAN, E.. **Introduction to online convex optimization**. Foundations and Trends® in Optimization, 2(3-4):157–325, 2016.

[40] ARORA, S.; HAZAN, E. ; KALE, S.. **The multiplicative weights update method: a meta-algorithm and applications**. Theory of Computing, 8(1):121–164, 2012.

[41] ZINKEVICH, M.. **Online convex programming and generalized infinitesimal gradient ascent**. In: PROCEEDINGS OF THE 20TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML-03), p. 928–936, 2003.

[42] SCHRIJVER, A.. **Theory of linear and integer programming**. John Wiley & Sons, 1998.

[43] PISINGER, D.. **Where are the hard knapsack problems?** Computers & Operations Research, 32(9):2271–2284, 2005.