Pedro Ferraz Villela

UM ALGORITMO EXATO PARA OBTER O CONJUNTO SOLUÇÃO DE PROBLEMAS DE PORTFÓLIO

CAMPINAS
2014

Universidade Estadual de Campinas

Instituto de Matemática, Estatística
e Computação Científica

Pedro Ferraz Villela

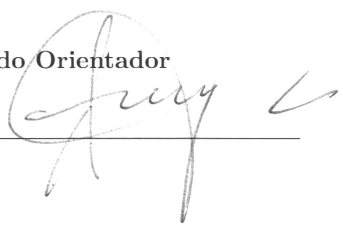
UM ALGORITMO EXATO PARA OBTER O CONJUNTO SOLUÇÃO DE PROBLEMAS DE PORTFÓLIO

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em matemática aplicada.

Orientador: Francisco de Assis Magalhães Gomes Neto

Este exemplar corresponde à versão final da tese defendida pelo aluno Pedro Ferraz Villela, e orientada pelo Prof. Dr. Francisco de Assis Magalhães Gomes Neto.

Assinatura do Orientador



Campinas
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

V715a

Villela, Pedro Ferraz, 1982-
Um algoritmo exato para obter o conjunto solução de problemas de portfólio / Pedro Ferraz Villela. – Campinas, SP : [s.n.], 2014.

Orientador: Francisco de Assis Magalhães Gomes Neto.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

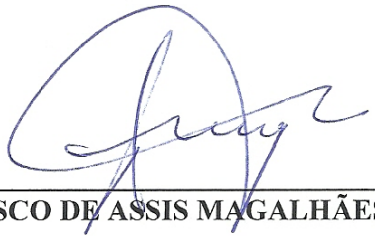
1. Otimização de carteiras de investimento. 2. Problema de otimização multiobjetivo . 3. Fronteira eficiente. 4. Método branch and bound. 5. Programação quadrática inteira mista. I. Gomes Neto, Francisco de Assis Magalhães,1964-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

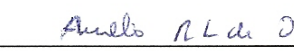
Título em outro idioma: An exact algorithm to obtain the solution set to portfolio problems
Palavras-chave em inglês:
Portfolio optimization
Multi-objective optimization problem
Efficient frontier
Branch and bound method
Mixed integer quadratic programming
Área de concentração: Matemática Aplicada
Titulação: Doutor em Matemática Aplicada
Banca examinadora:
Francisco de Assis Magalhães Gomes Neto [Orientador]
Aurelio Ribeiro Leite de Oliveira
Antonio Carlos Moretti
Luiz Leduíno de Salles Neto
Antônio Augusto Chaves
Data de defesa: 29-08-2014
Programa de Pós-Graduação: Matemática Aplicada

Tese de Doutorado defendida em 29 de agosto de 2014 e aprovada

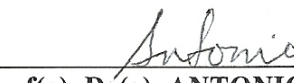
Pela Banca Examinadora composta pelos Profs. Drs.



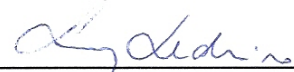
Prof(a). Dr(a). FRANCISCO DE ASSIS MAGALHÃES GOMES NETO



Prof(a). Dr(a). AURELIO RIBEIRO LEITE DE OLIVEIRA



Prof(a). Dr(a). ANTONIO CARLOS MORETTI



Prof(a). Dr(a). LUIZ LEDUÍNO DE SALLES NETO

Prof(a). Dr(a). ANTÔNIO AUGUSTO CHAVES

Abstract

In this work, we propose an exact method to find the solution set of a mixed quadratic bi-objective portfolio optimization problem. Our method is based on the combination of three specific algorithms. The first one obtains a curve associated with the solution set of a continuous bi-objective problem through an active set algorithm, the second one solves a mixed quadratic optimization problem through the Branch-and-Bound method, and the third one searches the intersection of two curves associated with distinct bi-objective problems. Throughout the text, some heuristics are also introduced in order to accelerate the performance of the method.

Moreover, our method can be seen as a new contribution to the field, since it finds, in an exact way, the curve related to the solution set of the mixed integer bi-objective problem, something uncommon in the corresponding literature, where the target problem is usually approached by metaheuristic methods. Additionally, it has also shown to be efficient in terms of running time, being capable of finding the problem's solution set within a much faster time frame.

Keywords: Multi-Objective Optimization. The Weighted Sum Method. Portfolio Optimization. Efficient Frontier. Active Set Method. Branch-and-Bound. Mixed Quadratic Programming.

Resumo

Neste trabalho, propomos um método exato para obter o conjunto solução de um problema biobjetivo quadrático de otimização de carteiras de investimento, que envolve variáveis binárias. Nosso algoritmo é baseado na junção de três algoritmos específicos. O primeiro encontra uma curva associada ao conjunto solução de problemas biobjetivo contínuos por meio de um método de restrições ativas, o segundo encontra o ótimo de um problema de programação quadrática

inteira mista pelo método *Branch-and-Bound*, e o terceiro encontra a interseção de duas curvas associadas a problemas biobjetivo distintos. Ao longo do texto, algumas heurísticas e métodos adicionais também são introduzidos, com o propósito de acelerar a convergência do algoritmo proposto.

Além disso, o nosso método pode ser visto como uma nova contribuição na área, pois ele determina, de forma exata, a curva associada ao conjunto solução do problemas biobjetivo inteiro misto, algo que é incomum na literatura, pois o problema alvo geralmente é abordado via métodos meta-heurísticos. Ademais, ele mostrou ser eficiente do ponto de vista do tempo computacional, pois encontra o conjunto solução do problema em poucos segundos.

Palavras chave: Programação Multiobjetivo. Método da Soma Ponderada. Otimização de Carteiras de Investimento. Fronteira Eficiente. Método de Restrições Ativas. *Branch-and-Bound*. Programação Quadrática Inteira Mista.

Sumário

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu orientador e amigo Chico, que sempre esteve junto comigo durante todo o trabalho. Agradeço por toda a sua disposição e paciência ao longo desse período.

Também agradeço muito aos meus pais, que se dispuseram a me ajudar até nos horários mais indesejáveis.

Por fim, agradeço ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), pelo essencial apoio financeiro prestado durante a vigência do trabalho.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Introdução

Um problema biobjetivo é aquele em que procuramos otimizar duas funções ao mesmo tempo, satisfazendo um conjunto de restrições. Em geral, a solução desse tipo de problema não é única, pois pode não existir uma solução que otimiza os dois objetivos simultaneamente. As soluções desse tipo de problema podem ser definidas pelo conceito de *dominância*. A curva no plano que representa todas as soluções desse tipo de problema é chamada de fronteira eficiente de Pareto.

Neste trabalho, estamos interessados em resolver problemas biobjetivo que envolvem, de forma simultânea, a minimização do risco e a maximização do retorno de carteiras de investimento, ou portfólios, que são os conjuntos de ativos financeiros nos quais se investe.

Contextualização do problema

Nos problemas que envolvem a otimização de carteiras de investimento, o investidor procura maximizar o lucro, bem como minimizar o risco de perda de capital, ao montar o seu portfólio. Para lidar com essa situação, utiliza-se um modelo matemático que leva em conta esses dois objetivos antagônicos.

Uma das primeiras formulações matemáticas para esse problema de investimento foi o modelo de Média-Variância proposto por Markowitz [30] nos anos 50 do século XX. Ainda hoje, esse modelo tem uma boa aceitação por parte dos investidores. Em sua forma usual, o modelo de Markowitz se resume a um problema de programação quadrática no qual a função objetivo representa o risco a ser minimizado e o retorno mínimo da carteira é descrito por uma restrição linear.

Com o passar do tempo, várias alterações foram sugeridas com o propósito de tornar o modelo de Markowitz mais compatível com o mercado financeiro real. Dentre essas alterações, destacamos a inclusão de restrições que limitam o tamanho da carteira e de restrições que estipulam um limite inferior para o montante aplicado em cada ativo, que podem ser formuladas a partir de variáveis binárias que controlam a existência de investimento em cada ativo, fazendo com que a carteira a ser montada varie de acordo com essas variáveis.

Com essas novas restrições, o problema de programação quadrática passa a envolver variáveis inteiras, além das variáveis reais que já eram usadas para representar o valor investido em cada ativo, de forma que, com essa inclusão, o problema passa a ser NP-difícil, geralmente exigindo um grande esforço computacional para a obtenção da solução ótima.

Em nosso trabalho, criarmos uma proposta para obter, de forma exata, o conjunto solução, assim como uma representação alternativa da fronteira eficiente de Pareto, de problemas de investimento biobjetivo que envolvem, tanto maximizar o retorno quanto minimizar o risco, de carteiras de investimento que satisfazem restrições lineares envolvendo variáveis binárias.

Objetivos e contribuições

A maior parte das contribuições feitas ao nosso problema na literatura envolve o emprego de meta-heurísticas, sendo raros os artigos que descrevem métodos exatos para resolvê-lo. Entretanto, existem métodos exatos que resolvem o problema quando a carteira está fixada tais como aquele proposto por Stein [35]. Da mesma forma, também são conhecidos métodos exatos tais como o proposto por Villela [36], que resolvem o problema quando combinamos os dois objetivos (minimização do risco e maximização do retorno) por meio de um peso (λ), cujos valores são discretizados ao longo de uma malha. Seguindo essas ideias, a proposta do nosso algoritmo geral consiste em combinar esses dois métodos com outro que percorre a fronteira de Pareto alternativa.

Nosso método, que começa pela extremidade direita da curva procurada, caminha pela curva de baixo (fronteira eficiente alternativa), variando o valor do peso que concilia os dois objetivos, até um ponto no qual pode haver mudança nos ativos da carteira. Quando isso ocorre, resolvemos o problema exato para o peso relacionado a essa mudança, utilizando uma combinação dos algoritmos de programação quadrática inteira mista, CHR [26], aliado a uma Heurística de melhoramento (uma de nossas contribuições), e ao *Branch-and-Bound* implícito [4]. Se a carteira ótima para o valor desse peso não é a mesma, procuramos o ponto da fronteira eficiente alternativa no qual há a interseção das curvas dessas duas carteiras, por meio da parametrização delas em formato de parábola (outra de nossas contribuições). Tal processo é repetido até determinarmos a extremidade esquerda da curva em questão.

Esta tese está dividida da seguinte forma: no primeiro capítulo, comentamos sobre problemas de programação multiobjetivo, enfatizando o conceito de fronteira eficiente de Pareto. O Capítulo 2 é dedicado a modelos de carteira de investimento e suas possíveis variações. No Capítulo 3, apresentamos o tipo de problema específico que nos propomos a resolver, enfatizando a combinação de três algoritmos específicos usada para resolvê-lo. No Capítulo 4, apresentamos o primeiro algoritmo

específico, que é utilizado para obter um segmento da a curva associada ao conjunto solução de problemas biobjetivo contínuos. Já no Capítulo 5, descrevemos o segundo algoritmo, que tem como objetivo encontrar a solução de problemas de programação quadrática inteira mista. Da mesma forma, no Capítulo 6, explicamos o terceiro algoritmo, que é utilizado para encontrar a interseção de duas curvas distintas associadas ao nosso problema original, assim como apresentamos o nosso algoritmo principal, foco de nossa proposta. O Capítulo 7 apresenta os resultados computacionais obtidos com a aplicação do nosso algoritmo a problemas reais de carteiras de investimento, que comprovam a eficiência do método que desenvolvemos. Finalmente, no Capítulo 8 comentamos os resultados obtidos, assim como possíveis trabalhos futuros.

Capítulo 1

Programação multiobjetivo

O problema a ser estudado nesse trabalho caracteriza-se em otimizar ao mesmo tempo dois ou mais objetivos, muitas vezes antagônicos, ao mesmo tempo. Situações desse tipo podem ser encontradas em diversas áreas do conhecimento humano, desde as mais simples, algumas aplicáveis ao nosso cotidiano, até aquelas mais sofisticadas, que envolvem produção industrial em larga escala.

Um exemplo clássico desse tipo de abordagem aparece no problema industrial que envolve o uso de diferentes padrões de corte na confecção de produtos. Nesse caso, temos como objetivo minimizar conjuntamente o desperdício e o custo de preparo das máquinas. Outra aplicação pode ser vista no planejamento de produtos que serão lançados no mercado, em que parte dos projetos visa tanto maximizar a eficiência da peça, quanto minimizar o custo de produção. Já outro exemplo desse tipo de problema pode ser visto na área de economia, quando o banco central procura estabelecer uma política monetária que tenta minimizar, ao mesmo tempo, a inflação, o desemprego e o balanço comercial. Finalmente, outra aplicação muito conhecida pode ser vista no ramo dos investimentos financeiros, quando procuramos maximizar o retorno de uma aplicação ao passo que minimizamos o risco da perda de capital.

Esses são alguns dos problemas de programação multiobjetivo. Em geral, procuramos minimizar um vetor de funções, em que cada uma representa um dos objetivos, sujeito a um conjunto de restrições. Na seção a seguir, vamos caracterizar matematicamente esse tipo de problema.

1.1 A solução de problemas multiobjetivo

Dados p objetivos distintos, cada um deles representado por uma função diferente, podemos escrever um problema geral de programação multiobjetivo da seguinte maneira:

$$\begin{array}{rl} \min & F(x) \\ S.a & G(x) \leq 0 \\ & H(x) = 0, \end{array} \tag{1.1}$$

onde

- $x \in \mathbb{R}^n$ é o vetor de variáveis.
- $F(x) = [f_1(x), f_2(x), ..., f_p(x)]^T$, representa o vetor de objetivos a ser minimizado.
- $G(x) = [g_1(x), g_2(x), ..., g_{m_I}(x)]^T$ representa o vetor de restrições de desigualdade.
- $H(x) = [h_1(x), h_2(x), ..., h_{m_E}(x)]^T$ representa o vetor de restrições de igualdade.

Em geral, não existe um único ponto x^* que minimiza todos os objetivos ao mesmo tempo, principalmente quando os objetivos são conflitantes, ou seja, esse tipo de problema não possui solução única. Por exemplo, o ponto x^j que minimiza $f_j(x)$ pode, ao mesmo tempo, maximizar $f_{j+1}(x)$. Dessa forma, precisamos determinar um critério para comparar duas soluções diferentes do Problema (1.1). Usualmente, essa comparação é feita por meio do conceito de *dominância*, inicialmente introduzido, no contexto de economia, por Pareto [33] em 1906. Dizemos que uma solução x^1 domina uma outra x^2 se:

- $\forall i : f_i(x^1) \leq f_i(x^2)$,
- $\exists i : f_i(x^1) < f_i(x^2)$.

Ou seja, uma solução domina a outra se ela é melhor em um dos objetivos e melhor ou igual nos restantes. Uma solução que não é dominada pelas demais é chamada de solução não dominada, ou ponto não dominado, ou solução eficiente, ou até mesmo de solução ótima de Pareto.

Como uma solução não dominada é sempre melhor que uma dominada, podemos dizer que o conjunto solução de (1.1) é composto por todos os pontos não dominados. Este conjunto é conhecido na literatura como *conjunto ótimo de Pareto*.

Quando $p \leq 3$, podemos visualizá-lo graficamente ao desenhar todas as soluções não dominadas em um sistema de coordenadas, em que cada eixo representa o valor de um dos objetivos. Quando

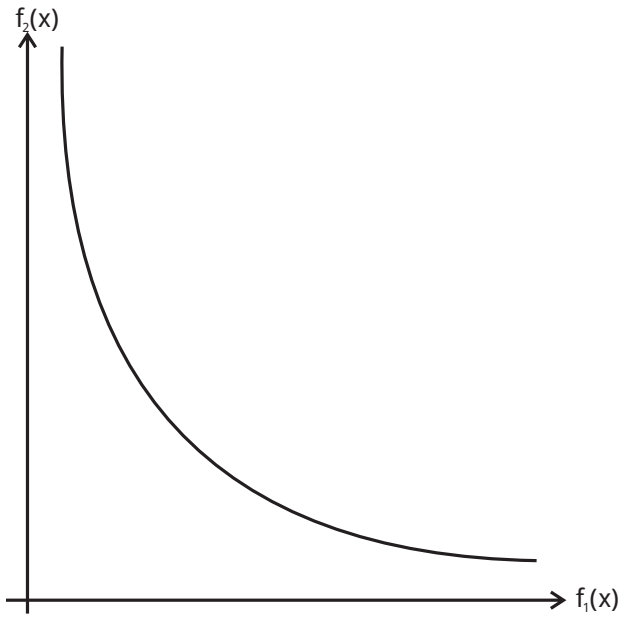


Figura 1.1: Fronteira eficiente de um problema biobjetivo.

$p = 2$, chamamos de Fronteira Eficiente de Pareto, ou Curva de Pareto, a curva composta por todos os pontos não dominados. A Figura 1.1 ilustra uma fronteira eficiente.

Uma abordagem clássica para problemas multiobjetivo, inicialmente proposta por Zadeh [37] em 1963, consiste em atribuir pesos distintos a cada um dos objetivos, de forma a minimizar a combinação convexa dos mesmos. Essa técnica é conhecida como o método da soma ponderada. Nesse caso, atribuímos pesos $\lambda_i \geq 0$, aos objetivos $f_i(x)$, de modo que $\sum_{i=1}^{p} \lambda_i = 1$, e resolvemos uma sequência de problemas de programação não linear do tipo

$$\begin{array}{rl} \min & \sum_{i=1}^{p} \lambda_i f_i(x) \\ S.a & G(x) \leq 0 \\ & H(x) = 0, \end{array} \tag{1.2}$$

variando o valor dos pesos da combinação convexa. Como cada solução do Subproblema (1.2) é um ponto não dominado, é possível encontrar um conjunto de pontos ótimos de Pareto discretizando os pesos λ_i no intervalo [0,1] e resolvendo o Subproblema (1.2) para cada um dos valores fixados. Entretanto, como existem p pesos diferentes, e apenas um deles pode ser escrito em função dos demais pela condição $\sum_{i=1}^{p} \lambda_i = 1$, temos que a ordem da direção de busca do problema original é reduzida em um e seria necessário trabalhar com uma malha de dimensão $p - 1$. Dessa forma, estaríamos trabalhando com a minimização conjunta sobre $p - 1$ parâmetros variantes, algo que,

na prática, pode vir a ser muito caro e difícil de se calcular.

No caso particular de problemas biobjetivo convexos, podemos escrever o peso do segundo objetivo em função do primeiro, produzindo uma sequência de problemas de objetivos únicos com apenas um parâmetro λ. A fronteira eficiente de problemas biobjetivo convexos costuma ser "bem-comportada", uma vez que ela é não crescente e possui reta suporte. Porém, mesmo nesse caso, os resultados obtidos pela discretização do valor de λ não são satisfatórios para a obtenção de uma representação precisa da fronteira eficiente, pois uma distribuição uniforme de valores de parâmetro não necessariamente gera uma distribuição uniforme de pontos ótimos de Pareto, fazendo com que alguns pontos importantes sejam perdidos. Dessa forma, para se obter resultados mais eficientes, é necessário combinar essa técnica com alguma outra que encontre os pesos mais relevantes a serem analisados, ou, talvez, trabalhar com uma técnica que não envolva esse processo. A estratégia de encontrar os pesos mais relevantes será utilizada nesse trabalho, conforme iremos mostrar no Capítulo 4.

Na seção a seguir, faremos uma revisão bibliográfica do assunto, comentando as diversas técnicas existentes para a resolução desse tipo de problema, assim como estratégias que possibilitam a redução do custo computacional.

1.2 Revisão literária

Dentre os diversos trabalhos na área de otimização multiobjetivo, optamos por revisar três tipos de linha de pesquisa: aquelas que utilizam o método da soma ponderada, aquelas que adaptam métodos clássicos de otimização para problemas multiobjetivo e aquelas que propõem algo novo, diferente do usual. Nas subseções a seguir, vamos descrever algumas contribuições feitas nessas três linhas de trabalho.

1.2.1 Trabalhos que envolvem o método da soma ponderada

Apesar de suas limitações, o método da soma ponderada é de fácil compreensão e implementação, sendo capaz de gerar bons resultados quando aliado a algum tipo de algoritmo específico, tanto que grande parte das contribuições literárias se baseiam nessa técnica.

Dentre elas, podemos destacar o trabalho de Kim e Weck [27], que propõe um método para resolver problemas biobjetivo por meio de uma modificação do método das somas ponderadas, mudando os pesos de forma adaptativa. Esse método, que é capaz de resolver problemas biobjetivo em regiões não convexas, mostrou potencial para ser expandido para problemas com mais de dois objetivos. A abordagem de soma adaptativa também aparece na contribuição de Eichfelder [17],

que foi capaz de encontrar pontos equidistantes da superfície de Pareto - que nada mais é do que a curva de Pareto tridimensional - de problemas convexos.

Uma outra contribuição muito importante é o trabalho de Das e Dennis [12], que apresenta um método, baseado na ideia da soma ponderada, capaz de resolver eficientemente problemas com mais de dois objetivos e traçar a maior parte da superfície de Pareto.

Outro trabalho importante que envolve a técnica da soma ponderada é o método do ϵ-restrito, proposto por Chankong e Haimes [10]. Esse método, que depois foi aperfeiçoado por Ehrgott e Ruzika [18], procura encontrar um ponto não dominado por meio da equivalência entre o problema multiobjetivo original e o problema ponderado segundo a técnica do ϵ-restrito. Essa técnica envolve resolver o problema original em um dos p objetivos, ao passo que os demais são restringidos por valores ϵ_i. Apesar de, na teoria, ser capaz de resolver boa parte dos problemas multiobjetivo, esse método não é muito eficiente do ponto de vista computacional, pois é necessária uma longa sequência de problemas de objetivos simples para se encontrar cada ponto não dominado.

Ainda, nessa linha de raciocínio, podemos destacar a contribuição de Plas *et al.* [34], que compararam o desempenho da técnica do ϵ-restrito, com o método da soma ponderada e dois algoritmos genéticos conhecidos na área, quando aplicados a problemas multiobjetivos que envolviam a alocação de instalações. Nessa pesquisa, os autores constataram que os métodos exatos superaram os heurísticos significativamente na categoria de problemas investigados, pois foram capazes de encontrar as fronteiras de *Pareto* com uma precisão muito maior.

Podemos destacar também a contribuição de Fliege e Heseler [22], que foram capazes de encontrar novos pontos ótimos de Pareto de problemas quadráticos convexos multiobjetivo, por meio do uso do processo de *warm start* sobre os pontos já conhecidos. Nesse trabalho, o método da soma ponderada gera uma sequência de problemas em que as condições KKT são suficientes para garantir a existência de um minimizador global. Esses problemas são resolvidos por meio do método de pontos interiores, e a técnica de *warm start* é utilizada para refinar a solução sempre que necessário. Esse método mostrou ser mais rápido que boa parte dos métodos usuais para problemas biobjetivo. Seguindo essas ideias, Fliege publicou outra contribuição nessa mesma linha de pesquisa, cujo conteúdo pode ser encontrado em [23]

Ainda, vale lembrar o estudo de Audet *et al.* [2], que procura formar a fronteira de Pareto de problemas biobjetivo caixa-preta, combinando a busca linear direta, com a discretização do domínio em forma de malhas e a teoria de Clarke [11]. Nesse método, procura-se resolver os subproblemas provenientes do método da soma ponderada por meio do algoritmo MADS (Multi Adaptative Direct Search), que consiste em realizar uma busca direta em cima da região factível discretizada na forma de uma malha.

A aplicação do método da soma ponderada a problemas biobjetivo também pode ser encontrada no trabalho de Stein [35], que desenvolveu uma técnica para encontrar os pesos mais representativos da fronteira eficiente de problemas de investimentos quadráticos, que são convexos por natureza. Para tanto, é utilizado um método de restrições ativas sobre o sistema KKT, resultante das condições de primeira ordem do problema ponderado.

Outra aplicação dessa técnica na área de investimentos financeiros pode ser encontrada no trabalho de Ehrgott e Waters [19], no qual é utilizada uma função de utilidade para medir as preferências do investidor, de modo que, a cada passo, resolvemos um problema escalarizado para determinar o portfólio que maximiza essa função.

Finalmente, podemos destacar os trabalhos de Cesarone *et al.* [7, 8], Chang *et al.* [9] e Dias [13], que foram capazes de aproximar a fronteira eficiente de problemas de investimentos biobjetivo que envolvem variáveis binárias, por meio da combinação da soma ponderada com meta-heurísticas.

1.2.2 Trabalhos que envolvem a adaptação de técnicas clássicas de otimização

Para evitar as falhas intrínsecas da técnica da soma ponderada, muitas das contribuições literárias tentam encontrar o conjunto ótimo de Pareto por meio da adaptação de algoritmos clássicos de otimização, tais como o método de Newton e o algoritmo *Branch-and-Bound*, para problemas multiobjetivo.

Dentre elas, podemos destacar o trabalho de Fliege *et al.* [21], que encontra o conjunto ótimo de Pareto partindo de diferentes pontos aleatórios de problemas irrestritos, que envolvem funções que são duas vezes continuamente diferenciáveis. Para isso, é feita uma adaptação do método de Newton e da busca linear de Armijo (vide página 281 de [3]) para problemas multiobjetivo. Essa abordagem foi posteriormente estendida para problemas mais complexos, como podemos ver na contribuição de Drummond e Sveiter [16].

Seguindo a linha de adaptação de métodos clássicos, outra contribuição importante de Fliege e Sveiter [24] mostra como encontrar um ponto não dominado de problemas irrestritos, ou com restrições Lipshitz contínuas, por um método de direções de descida. A ideia central do trabalho é criar um método que convirja para um ponto não dominado no caso irrestrito e ache uma direção de descida factível, no caso restrito. No caso irrestrito, a direção de descida é encontrada ao se resolver um subproblema com um único objetivo, enquanto o tamanho do passo é obtido por meio de uma busca que usa uma regra tipo Armijo.

Outra adaptação de um método clássico bastante conhecido pode ser vista na contribuição de

Drummond e Iusem [14], na qual eles utilizaram uma versão modificada do método do Gradiente projetado para encontrar soluções de Pareto fracamente eficientes[1] em espaços convexos de dimensão finita. Esse trabalho foi posteriormente aperfeiçoado por Drummond e Fukuda em [15] que foram capazes de mostrar que, sob a condição dos objetivos serem convexos, o algoritmo converge independentemente da escolha do ponto inicial.

Uma outra contribuição que utiliza a adaptação de técnicas clássicas pode ser encontrada no trabalho de Fernández e Tóth [20], que utiliza o método *Branch-and-Bound* intervalar para formar a fronteira eficiente de Pareto de problemas biobjetivo não lineares. Nesse caso, o método *Branch-and-Bound* intervalar, que pode ser entendido como uma extensão do método *Branch-and-Bound* comum, é utilizado para formar a fronteira de Pareto por meio de testes de descarte, similares àqueles utilizados no método convencional. Apesar de promissor, esse método é muito lento se usado de forma pura, sem nenhuma heurística para acelerá-lo.

Também podemos destacar a contribuição de Leyffer [28], que utiliza programação por metas, programação em dois níveis e restrições de complementaridade para encontrar o conjunto ótimo de Pareto de problemas convexos multiobjetivo. Nesse trabalho, procura-se encontrar os pontos não dominados da forma mais uniforme possível, por meio de um parâmetro de uniformidade que é adicionado ao problema modificado a ser resolvido. Dessa forma, primeiramente transformamos o problema de programação multiobjetivo original em um problema de programação em dois níveis, para depois transformarmos as restrições de segundo nível em restrições de cardinalidade.

Finalmente, o trabalho de Antczak [1] procura encontrar pontos ótimos de Pareto de problemas multiobjetivo não lineares que envolvem restrições de desigualdade, utilizando conceitos como a função lagrangeana, a invexidade e a qualificação de restrições. Para tanto, sob algumas hipóteses específicas tais como a invexidade, é possível transformar o problema multiobjetivo original em um problema modificado equivalente. A grande vantagem do problema modificado, em relação ao original, é que, sob algumas hipóteses, ele pode se tornar linear, sendo assim mais tratável.

1.2.3 Trabalhos que envolvem técnicas não usuais

Existem algumas contribuições literárias que utilizam técnicas pouco convencionais. Dentre elas, podemos destacar o trabalho de Mirttinen e Makela [31] que, por meio da teoria de cones, procura encontrar soluções de Pareto, tanto as eficientes quanto as fracamente eficientes, de problemas multiobjetivo, que envolvem objetivos não lineares. Nesse trabalho, procura-se caracterizar os ótimos de Pareto, tanto do caso convexo, quanto do caso não convexo, por meio dos conceitos de

[1] São soluções eficientes que exercem dominância de forma estrita em todos os objetivos.

cone tangente e cone normal. Esse trabalho, que assume uma abordagem completamente teórica, não apresenta resultados computacionais.

Dentre todas as contribuições destacadas, aquelas que pareceram mais promissoras foram as de Fliege *et al.* [21], Kim e Weck [27] e Stein [35]. Optamos, inclusive, por incorporar a última delas em nosso trabalho, conforme veremos no Capítulo 4.

Capítulo 2

Modelos de carteiras de investimento

2.1 O modelo de Markowitz

O tipo de problema de portfólio a ser estudado nesse trabalho caracteriza-se pela escolha de como investir o capital disponível entre as várias ações existentes no mercado. A estratégia de resolução desse tipo de problema envolve tanto a maximização do lucro do investimento, como a minimização do risco de perda de capital em virtude das flutuações dos preços do mercado. Naturalmente, estes objetivos são contraditórios e cabe ao investidor decidir como combiná-los.

O modelo matemático clássico para esse tipo de problema de portfólio é baseado nas ideias propostas por Markowitz [30] em 1952. Definidos os n diferentes ativos que compõem a carteira de investimento, o montante a ser investido em cada ativo pode ser obtido resolvendo-se o problema de programação quadrática

$$\begin{array}{ll} \min & \frac{1}{2}x^T Q x \\ S.a & \mu^T x = \rho \\ & \sum\limits_{i=1}^{n} x_i = 1 \\ & x_i \geq 0, \qquad i = 1,...,n, \end{array} \tag{2.1}$$

onde

- $Q \in \mathbb{R}^{nxn}$ é a matriz de covariância do problema. O elemento (i,j) dessa matriz representa a covariância, que é a medida de quanto duas variáveis aleatórias se influenciam, do ativo i em relação ao ativo j. Além disso, a matriz Q é simétrica e semi-definida positiva, ou seja, $Q = Q^T$ e $x^T Q x \geq 0$, $\forall x \in \mathbb{R}^n$.

- $x \in \mathbb{R}^n$ é o vetor de incógnitas, que indica como o montante total disponível deve ser

distribuído entre os ativos. Desta forma, a componente x_i fornece a fração do montante disponível a ser investida no ativo i.

- $\mu \in \mathbb{R}^n$ é o vetor de retornos esperados. Ou seja, o elemento μ_i representa o retorno esperado para o ativo i.
- $\rho \in \mathbb{R}$ é o nível desejado de retorno do portfólio.

Nesse modelo, temos como objetivo minimizar o risco, representado pela função quadrática baseada na matriz de covariância, para um nível predefinido de retorno, definido pela primeira restrição. A segunda restrição indica que desejamos investir todo o montante disponível, enquanto a terceira é usada para evitar que a porcentagem do dinheiro investida em cada ativo seja negativa, o que significa que não permitimos a venda de ativos.

Para definir um problema específico na forma (2.1), faz-se uma análise histórica dos ativos financeiros escolhidos para compor a carteira, de modo que seja possível criar tanto uma medida de risco (em geral, a variância), como também obter o valor esperado de retorno de capital (esperança).

2.2 Outros modelos

O Modelo (2.1) poderia ser substituído por outro no qual se estabelecesse um nível para a variância e o retorno fosse maximizado. Outra possibilidade seria a adoção de uma desigualdade na restrição de retorno esperado, de modo que ρ correspondesse ao retorno mínimo admissível. Ao longo dessa seção, apresentaremos outros modelos que podem ser obtidos a partir de (2.1).

2.2.1 O modelo paramétrico

Uma modificação possível do Modelo (2.1) seria incluir o objetivo de maximizar o retorno na função objetivo. Para tanto, é preciso utilizar um parâmetro λ que pondera os objetivos conflitantes de redução do risco e aumento do retorno. Neste caso, obtemos o problema abaixo, no qual temos apenas as restrições de uso de todo o montante disponível e de não negatividade das variáveis.

$$\begin{array}{ll} \min & \frac{1}{2}\lambda x^T Q x - (1-\lambda)\mu^t x \\ S.a & \sum\limits_{i=1}^{n} x_i = 1 \\ & x_i \geq 0, \qquad\qquad i = 1,...,n. \end{array} \tag{2.2}$$

Tomando $0 \leq \lambda \leq 1$, observamos que, para $\lambda = 0$, apenas o retorno é considerado, enquanto $\lambda = 1$ corresponde à simples minimização do risco.

Supondo que a matriz Q seja positiva definida, temos que, para cada valor fixo de λ, o problema é convexo e possui solução única, pois o único minimizador local existente é também minimizador global.

Uma variação do Modelo (2.2) consiste em escalar a função objetivo, de forma que o parâmetro do risco seja sempre 1 e o parâmetro do retorno seja λ. Dessa forma, temos o seguinte modelo:

$$\begin{array}{ll} \min & \frac{1}{2}x^T Q x - \lambda \mu^T x \\ S.a & \sum_{i=1}^{n} x_i = 1 \\ & x_i \geq 0, \qquad\qquad i = 1,...,n. \end{array} \tag{2.3}$$

Nesse caso, se $\lambda = 0$ apenas o risco é considerado, enquanto $\lambda \to +\infty$ corresponde à simples maximização do retorno.

2.2.2 O modelo com variáveis binárias

Outra linha de desenvolvimento de modelos para a otimização de carteiras de investimentos está relacionada à inclusão de restrições que tornam o problema mais compatível com as situações encontradas na vida real. Nessa linha, destacamos o modelo no qual se restringe a quantidade de ativos que irão compor a carteira, uma vez que normalmente não é viável investir em todos os n ativos disponíveis no mercado. Este modelo geralmente inclui também um conjunto de limites inferiores, l_i, e superiores, u_i, para as frações aplicadas a cada ativo selecionado, pois, na prática, é comum existir uma quantia mínima a ser aplicada em um ativo que faz parte da carteira.

Com o intuito de restringir a K o número de ativos, incluímos no modelo as variáveis binárias auxiliares z_i , $i = 1,...,n$, de modo que $z_i = 1$ caso o ativo i faça parte da carteira e $z_i = 0$ caso contrário.

Já com relação aos limites inferiores e superiores, como estes só são aplicados às variáveis efetivamente incluídas na carteira, em lugar de utilizarmos canalizações simples, recorremos a desigualdades mistas, ou seja, com variáveis inteiras e reais.

Tomando por base o modelo de Markowitz, obtemos o seguinte problema

$$\begin{array}{rl} \min & \frac{1}{2}x^T Q x \\ S.a & \mu^t x = \rho \\ & \sum\limits_{i=1}^{n} x_i = 1 \\ & l_i z_i \leq x_i \leq u_i z_i, \quad i = 1,...,n \\ & z_i \in \{0,1\}, \qquad\quad i = 1,...,n \\ & \sum\limits_{i=1}^{n} z_i \leq K. \end{array} \tag{2.4}$$

Em (2.4), se z_i valer 0, x_i também será igual a 0. Por outro lado, se $z_i = 1$, temos $l_i \leq x_i \leq u_i$. Como temos no máximo K variáveis inteiras z_i iguais a 1, somos forçados a escolher no máximo K ativos.

Naturalmente, as restrições que envolvem as variáveis binárias podem ser incluídas em todos os problemas apresentados neste capítulo, bem como em muitos outros modelos para problemas de otimização de carteiras. Para mais detalhes, sugere-se consultar o artigo de Mitra *et al.* [32].

2.3 A fronteira eficiente de um problema de investimento

Problemas de programação multiobjetivo que envolvem risco e retorno não possuem uma solução que minimiza o primeiro e maximiza o segundo ao mesmo tempo, pois esses objetivos são conflitantes.

Sendo assim, no caso de aplicações financeiras, dizemos que um portfólio π_1 domina outro π_2 se π_1 possui risco menor e retorno maior que π_2, podendo haver igualdade em uma das duas medidas. Para esse problema biobjetivo, a fronteira eficiente de investimento (que chamaremos de FEI), ou simplesmente de fronteira de Pareto, é a curva não decrescente que contém as melhores possibilidades de conciliação entre risco e retorno.

A Figura 2.1, apresenta uma fronteira eficiente de investimento gerada a partir de ativos da bolsa de valores do Reino Unido. Nesta figura, o eixo das abscissas representa a medida de risco do portfólio, que nesse caso é a variância. Já o eixo das ordenadas representa o retorno médio dos ativos.

Vale notar que, caso não tenhamos as restrições que envolvem as variáveis binárias, a curva será "bem comportada", pois ela é não decrescente e possui uma reta suporte.

A análise da Figura 2.1 revela que, caso o investidor tenha uma grande aversão ao risco, ele deve optar por investir em um portfólio que tenha vários ativos, pois a possibilidade de perder

Figura 2.1: FEI dos ativos da bolsa de valores do Reino Unido (FTSE).

dinheiro em todos eles é menor. Por outro lado, caso o investidor seja mais audaz e escolha priorizar o retorno, ele deve optar por investir em poucos ativos, em geral aqueles que apresentam o maior retorno, ainda que o risco de perda de capital seja grande devido à volatilidade do mercado financeiro. Vale notar, ainda, que nenhuma das soluções da fronteira eficiente é "melhor" que as outras: cada ponto dela representa a melhor solução para o nível de risco especificado pelo investidor. Para mais detalhes sobre fronteira eficiente de investimentos, sugere-se consultar o livro de Luenberger [29].

Em geral, podemos determinar a fronteira eficiente de investimento por meio de modelos parametrizados tais como (2.2) e (2.3). Como a função objetivo desses problemas é convexa, todo mínimo local é também mínimo global. Logo, cada ponto da fronteira nada mais é que a solução do modelo para um ou mais valores de λ. Assim, o parâmetro λ desempenha um papel essencial, uma vez que representa o nível de risco e retorno escolhidos pelo investidor.

Uma maneira prática de se construir a fronteira eficiente passa pelo uso do método da soma ponderada, apresentado na Seção 1.1, que envolve a discretização dos valores λ no intervalo [0,1] e a resolução de (2.2) para cada um dos valores de λ selecionados. Em geral, essa discretização é feita por meio da divisão desse intervalo em subintervalos de comprimentos iguais, $\Delta\lambda$, de modo que $\lambda \in \{0,\Delta\lambda,2\Delta\lambda,...,1\}$. Um processo análogo poderia ser adotado para o Modelo (2.3), com a diferença de que, nesse caso, deveríamos discretizar o intervalo $[0,\lambda_{max}]$, em que λ_{max} é um número muito grande.

Como, para cada valor de λ, há uma ou mais soluções com o mesmo valor de função objetivo, podemos dizer que a curva bidimensional $\lambda \times f(\lambda)$, onde $f(\lambda)$ é o valor ótimo da função objetivo do problema paramétrico para um valor fixo de λ, é uma representação alternativa para a fronteira de Pareto. Essa curva é monótona da mesma forma que aquela da Figura 2.1, porém é decrescente. A Figura 2.2 representa a curva $\lambda \times f(\lambda)$ característica do Problema (2.2).

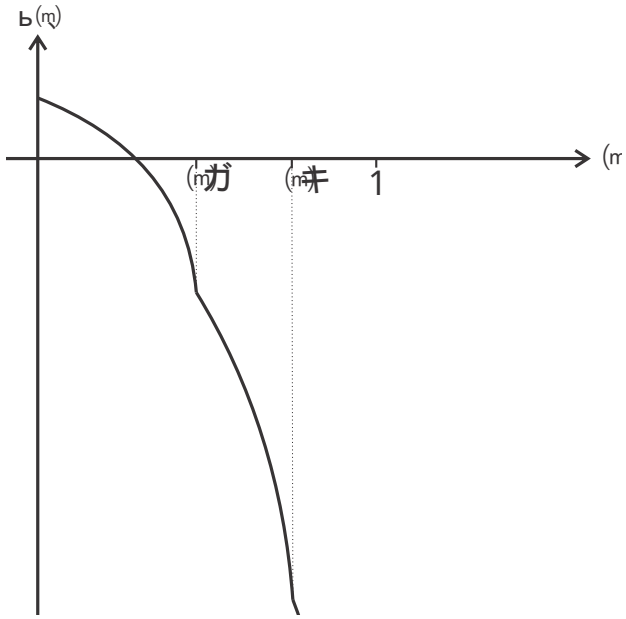


Figura 2.2: Curva $\lambda \times f(\lambda)$ do Problema (2.2).

As duas formulações apresentadas da fronteira eficiente de investimentos são equivalentes, pois ambas contêm as informações primordiais ao investidor. Isso se deve ao fato de que cada valor de λ da curva alternativa está associado a pelo menos um vetor x no espaço de variáveis originais, que seria exatamente uma solução não dominada da fronteira eficiente original.

Quando se incorpora as restrições que envolvem as variáveis binárias, a dificuldade de se aproximar a fronteira eficiente aumenta devido à constante troca dos ativos que compõem a carteira. Assim, a complexidade do Problema (2.4) é muito maior do que a de (2.2). Para mais informações sobre a fronteira eficiente em problemas que envolvem as restrições com variáveis binárias, pode-se consultar Chang *et al.* [9] e Dias [13].

2.4 Problemas de programação quadrática inteira mista

Um modelo mais próximo da realidade, pode ser criado combinando o modelo paramétrico (2.3) com o modelo que inclui restrições que envolvem variáveis binárias (2.4). Tal processo é

equivalente a acrescentar o retorno parametrizado à função objetivo do Problema (2.4). Para facilitar a resolução do problema, também é possível transformar em desigualdade a restrição de montante aplicado. O modelo obtido assim é o seguinte para $\lambda \in [0, +\infty)$:

$$\begin{array}{ll} \min & \frac{1}{2}x^T Q x - \lambda \mu^T x \\ S.a & \sum\limits_{i=1}^{n} x_i \leq 1 \\ & l_i z_i \leq x_i \leq u_i z_i, \quad i = 1,...,n \\ & z_i \in \{0,1\}, \qquad i = 1,...,n \\ & \sum\limits_{i=1}^{n} z_i \leq K. \end{array} \tag{2.5}$$

Com a conversão da restrição de montante aplicado em uma desigualdade, é possível que uma parte do dinheiro disponível não seja investida, particularmente no caso em que λ está próximo de 0, ou seja, quando a aversão ao risco é grande. Em particular, quando $\lambda = 0$, e a matriz Q é definida positiva, nada é investido. Neste caso, a função objetivo atingirá o mínimo apenas para $x = 0$.

Como não é razoável supor que um investidor mantenha em caixa grande parte de seu dinheiro, podemos considerar a existência de um ativo livre de risco (como a caderneta de poupança ou as letras do tesouro americano) que absorva o montante não aplicado no conjunto de ativos que pode compor a carteira.

Esse ativo livre de risco pode, inclusive, ser introduzido no Modelo (2.5), o que tornaria semi-definida positiva a matriz Q. Com essa alteração, todo o dinheiro disponível seria aplicado, exceto no caso em que $\lambda = 0$, situação para a qual há infinitas soluções ótimas, pois qualquer montante investido no ativo livre de risco seria ótimo, desde que nada fosse aplicado nos demais ativos. A introdução de um ativo livre de risco altera a fronteira eficiente, fazendo com que esta toque o eixo vertical apresentado na Figura 2.1.

Esse modelo foi tomado como base para o estudo realizado neste trabalho, conforme veremos no Capítulo 3.

Existe uma formulação alternativa do Problema (2.5) em que trabalhamos com a versão implícita das restrições dese modelo. Para entender como isso é feito, primeiramente devemos perceber que limitar o número de ativos na carteira nada mais é do que exigir que não mais que K componentes do vetor x sejam positivas. Logo, chamando de $nz(x) = \{i | x_i \neq 0\}$ o conjunto das componentes não nulas do vetor x, podemos substituir a restrição $\sum\limits_{i=1}^{n} z_i \leq K$ pela exigência de que a cardinalidade de $nz(x)$ seja, no máximo, K, ou seja, que $Card(nz(x)) \leq K$. Além disso, para manter as restrições de limite inferior, assim como as de positividade, basta exigir que as variáveis

cujos índices estão em $nz(x)$ satisfaçam as restrições de limite inferior, bem como que as demais sejam iguais a zero. Sendo assim, considerando que o limite superior das variáveis seja sempre 1, podemos reescrever o Problema (2.5) na forma

$$\begin{array}{rl} \min & \frac{1}{2}x^TQx - \lambda\mu^Tx \\ S.a & \sum\limits_{i=1}^{n} x_i \leq 1 \\ & Card(nz(x)) \leq K \\ & x_i \geq l_i, i \in nz(x) \\ & x_i = 0, i \notin nz(x). \end{array} \tag{2.6}$$

É fácil perceber que, caso o limitante superior das variáveis de (2.5) seja sempre 1, as restrições de (2.5) e (2.6) serão equivalentes, fazendo com que a solução dos dois problemas seja a mesma. Entretanto, a grande vantagem da formulação (2.6), em relação à formulação (2.5), é que podemos encontrar sua solução, para um valor fixado de λ, por meio de uma adaptação do método *Branch-and-Bound* que considera as restrições implícitas de forma eficiente. Essa adaptação, que será explicada cuidadosamente no Capítulo 5, foi utilizada em nosso trabalho para resolver uma classe de subproblemas que envolvem essa formulação.

Capítulo 3

Análise da fronteira eficiente alternativa

Problemas de portfólio e de programação multiobjetivo são muito explorados nos dias de hoje, sendo aqueles que envolvem risco e retorno uma das possíveis interseções entre essas duas áreas. Dentre eles, aquele que possui variáveis binárias e reais é um dos mais complexos. Essa complexidade aparece tanto no momento de resolver o problema com o parâmetro λ fixo, quanto na construção da fronteira de Pareto.

Porém, é possível construir a FEI alternativa desse tipo de problema tomando como base apenas os pontos chave, que são tanto aqueles em que ocorre a mudança de ativos na carteira (as variáveis binárias mudam de valor), quanto aqueles em que há uma mudança significativa na inclinação da curva, conforme veremos ao longo desse trabalho. Nesse capítulo, vamos mostrar parte desse processo ao apresentar uma estratégia para encontrar esses pontos, por meio da divisão do problema original em uma sequência de subproblemas, cujas curvas $\lambda \times f(\lambda)$ estão bem definidas.

3.1 A nossa proposta

A obtenção do conjunto solução de problemas risco-retorno do tipo (2.5) é uma área pouco explorada na literatura e, quando abordada, em geral a solução é obtida por meio de métodos metaheurísticos. Nesse trabalho, apresentamos um método direto eficiente para resolver problemas de portfólio de grande porte que envolvem variáveis binárias, com objetivo de encontrar a sua solução para todos os possíveis valores do parâmetro λ.

Em nossa proposta, nos concentramos no Modelo (2.5), que inclui uma restrição que limita o tamanho da carteira. Como, em geral, devido aos custos de transação, o investidor está mais preocupado com o mínimo a ser investido e não com o máximo, incluímos limitantes inferiores pré-definidos para os montantes a serem aplicados nos ativos da carteira, ao passo que o limitante

superior de cada um deles é sempre 1, ou seja, podemos investir 100% do nosso capital em um único ativo, desde que o mesmo faça parte da carteira. Dessa forma, o objetivo é encontrar a fronteira eficiente de investimentos alternativa do seguinte problema:

$$\begin{array}{lll} \min & f(x) = \frac{1}{2}x^T Q x - \lambda \mu^T x & \\ S.a & \sum\limits_{i=1}^{n} x_i \leq 1 & \\ & l_i z_i \leq x_i \leq z_i, & i = 1,...,n \\ & z_i \in \{0,1\}, & i = 1,...,n \\ & \sum\limits_{i=1}^{n} z_i \leq K, & \end{array} \tag{3.1}$$

onde $\lambda \in [0, +\infty)$, $K \leq n$ e Q é simétrica definida positiva. A complexidade de encontrar a FEI alternativa desse problema é enorme devido às variáveis binárias. Para cada valor de λ fixo, temos um problema de programação quadrática inteira mista, que não é fácil de se resolver. Além disso, a entrada ou a saída de um ativo da carteira, algo que ocorre com frequência quando variamos λ, provoca uma mudança drástica na inclinação da FEI alternativa.

Ao resolvermos o problema com as restrições de integralidade relaxadas, ou seja, admitindo que $z_i \in [0,1]$, notamos que, frequentemente, para valores grandes de λ, as variáveis z_i, de fato, valem 0 ou 1. Porém, para valores pequenos de λ, as variáveis z_i assumem valores fracionários. Tal resultado é compatível com a nossa intuição, pois quanto menor o valor de λ, maior é o risco, assim como a necessidade de diversificar o portfólio, o que nos leva a escolher um número elevado de ativos. Assim, a presença de variáveis fracionárias permite que, na prática, o limite K de ativos na carteira seja violado.

Como dito na Seção 2.3, uma estratégia simples para aproximar a FEI alternativa consiste em discretizar o intervalo $[0,\lambda_{max})$, em que λ_{max} é um número muito grande, e, para cada valor fixo de λ nesse intervalo, resolver o Problema (3.1). Esse problema, apesar de complexo, pode ser resolvido por métodos tais como o apresentado por Villela [36]. Entretanto, essa estratégia não permite que representemos a FEI alternativa do Problema (3.1) com uma precisão adequada, devido ao fato de não ser possível determinar, de forma exata, os pontos chave da mesma. Para compensar esse problema, seria necessário aumentar o tamanho da malha da discretização, o que, sem dúvida, tornaria o método caro computacionalmente. De qualquer forma, essa abordagem foi utilizada em nossos testes computacionais como parâmetro de comparação com o nosso algoritmo, conforme veremos no Capítulo 7.

3.2 Decompondo o problema

A vantagem de trabalhar com problemas que envolvem variáveis binárias é que podemos enumerar de forma finita todas as possíveis combinações de seus valores, obtendo assim todas as possíveis soluções do problema. Apesar dessa estratégia ser cara e ineficiente na prática, ela é útil para compreendermos de forma geral o comportamento do problema.

No caso do Problema (3.1), fixar as variáveis binárias z_i em zero ou um significa determinar se um ativo vai ou não compor a carteira. Sendo assim, a fixação das variáveis binárias vai resultar em um subproblema com variáveis contínuas x_i, que são as porcentagens a serem investidas nos ativos cuja variável binária correspondente tenha valor um.

Para ilustrar esse processo, vamos considerar um problema do tipo (3.1) com 3 ativos ($n = 3$), dos quais devem ser escolhidos no máximo 2 ($K = 2$). Nesse caso, podemos escolher para compor a carteira, nenhum ativo, ou somente o ativo 1, ou somente o ativo 2, ou somente o ativo 3, ou os ativos 1 e 2, ou os ativos 1 e 3, ou os ativos 2 e 3, totalizando $C_{3,0} + C_{3,1} + C_{3,2} = 1 + 3 + 3 = 7$ possibilidades.

De forma geral, para um problema com n ativos, o número total de subproblemas é $T = C_{n,0} + C_{n,1} + C_{n,2} + ... + C_{n,K}$, ou seja, o número de subproblemas cresce rapidamente conforme o número de ativos também cresce. Sendo assim, é inviável resolver todos eles na prática.

De qualquer maneira, o problema resultante para os ativos que estão fixados na carteira tem a seguinte forma:

$$\begin{aligned} \min \quad & f(x) = \tfrac{1}{2} x^T Q x - \lambda \mu^T x \\ S.a \quad & \sum_{i=1}^{n} x_i \leq 1 \\ & x_i \geq l_i, i \in I(z) \\ & x_i = 0, i \notin I(z), \end{aligned} \tag{3.2}$$

onde $I(z) = \{j, | z_j = 1\}$.

3.3 A fronteira eficiente com ativos fixados

O Subproblema (3.2) é similar ao Problema (2.3), com diferença no valor do limite inferior e na restrição sobre o montante máximo, que agora se tornou de desigualdade. Logo, é natural esperar que a FEI de ambos os problemas sejam similares. Devido a essa similaridade, é possível obter-se a FEI alternativa de (3.2) por meio de um processo similar ao descrito na Seção 2.3.

Ao longo desse trabalho, trabalharemos apenas com a formulação alternativa da FEI, ou seja,

a curva $\lambda \times f(\lambda)$, de modo que, a partir desse momento, vamos nos referir a ela como Fronteira Eficiente de Investimentos (FEI) ou apenas fronteira eficiente. A Figura 3.1 mostra uma FEI típica do Problema (3.2).

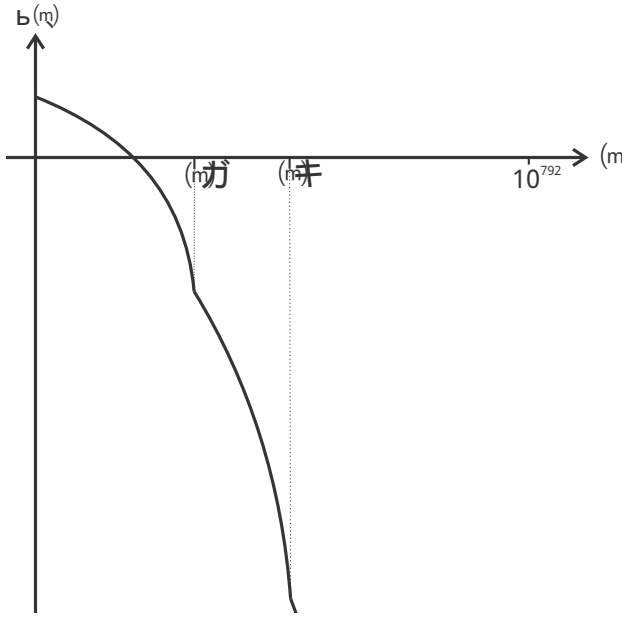


Figura 3.1: FEI do Subproblema (3.2).

Conforme pode-se observar no desenho, a FEI é uma curva monótona decrescente que pode ser representada pela união finita de segmentos de parábolas. Note que cada parábola está atrelada a intervalos específicos de λ.

3.4 Trabalhando com todas as curvas

Para um valor fixo de λ, o Problema (3.1) se torna um problema de programação quadrática inteira mista, ao passo que o Subproblema (3.2) é um problema de programação quadrática contínua. Como os T subproblemas contêm todas as combinações possíveis das variáveis binárias, a solução de (3.1) está contida no conjunto das soluções dos subproblemas do tipo (3.2). Nesse caso, como o Problema (3.1) é de minimização, a sua solução corresponderá à solução do subproblema do tipo (3.2) que possui o menor valor da função objetivo. Assim, podemos resolver todos os subproblemas e encontrar aquele que possui o menor valor da função objetivo.

Esse raciocínio também seria válido caso estivéssemos procurando a FEI de (3.1). Nessa situação, a FEI desse problema envolveria as T curvas associadas aos subproblemas com variáveis contínuas do tipo (3.2). Como a função objetivo de (3.1) é de minimização, podemos dizer que sua

FEI é a curva que está por baixo do conjunto de todas as T FEI associadas ao Subproblema (3.2). Logo, uma maneira de se obter a FEI de (3.1) seria desenhar, em um mesmo sistema de eixos, todas as curvas associadas ao Subproblema (3.2), e encontrar a curva que está por baixo delas, ou seja, achar a união dos segmentos de curvas que estão por baixo dos demais.

Para exemplificar essa situação, considere um problema do tipo (3.1) com três variáveis, em que temos que escolher exatamente duas para compor a nossa carteira, ou seja, a restrição sobre o número de ativos que vão compor a carteira é de igualdade. Nesse caso, teríamos três subproblemas do tipo (3.2) com curvas associadas C_1, C_2 e C_3. A curva C_1 seria correspondente ao subproblema em que os ativos 1 e 2 estão na carteira, ao passo que a curva C_2 considera que os ativos 1 e 3 estão na carteira, enquanto a curva C_3 considera que os ativos 2 e 3 estão na carteira. A Figura 3.2 mostra o desenho das 3 curvas em um mesmo sistema de eixos.

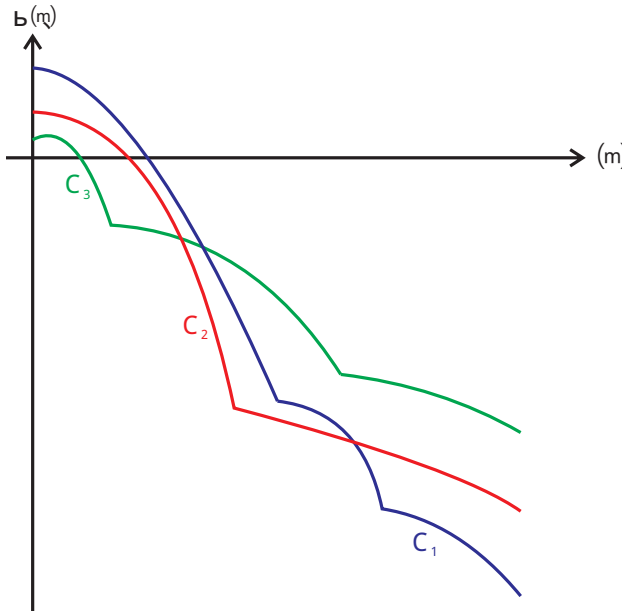


Figura 3.2: FEI dos subproblemas em um mesmo sistema de eixos.

A análise da Figura 3.2 nos revela que a curva que está por baixo possui segmentos das três curvas. Da esquerda para a direita, a curva que está por baixo começa com um segmento de C_3, depois continua com um segmento de C_2, mudando posteriormente para a curva C_1. A curva que está por baixo, ou seja, a FEI do Problema (3.1) desse exemplo, é mostrada na Figura 3.3. Dessa última figura, é possível notar que a FEI é contínua, apesar de sua derivada conter descontinuidades, provenientes da troca da curva que está por baixo ou da mudança de segmento de parábola.

Como se observa, é possível encontrar a FEI do Problema (3.1) por construção, a partir de uma sequência de FEI de Subproblemas (3.2). A ideia desse processo é dividir um problema que

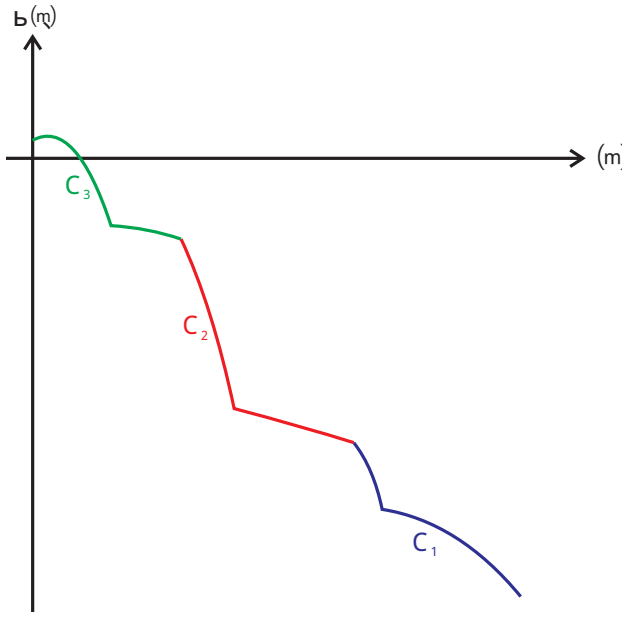


Figura 3.3: FEI do Problema (3.1).

possui variáveis binárias, com o qual é difícil de se trabalhar, em uma sequência de problemas com variáveis contínuas, que são de mais fácil tratamento.

3.5 Criando um novo método

O processo apresentado na seção anterior, apesar de ser esclarecedor do ponto de vista teórico, é muito caro computacionalmente, devido ao fato do número de subproblemas do tipo (3.2) crescer rapidamente conforme aumentamos o número de variáveis.

Além disso, as Figuras 3.2 e 3.3 deixam claro que diversos segmentos das três curvas não são aproveitados na curva final. Logo, seria mais conveniente caminhar na curva que está por baixo de forma inteligente, sem ter que, de fato, encontrar todas as curvas associadas à FEI do Problema (3.1). Tomando esse raciocínio como base, a proposta desse trabalho é, por meio da estrutura especial desse problema, criar um algoritmo eficiente para aproximar essa curva.

Para motivar o leitor, vamos explicar, em linhas gerais, como funciona o nosso algoritmo, tomando como base a Figura 3.4, que apresenta o exemplo da Figura 3.2, acrescido de seus respectivos pontos chave.

Nosso método sempre começa da extremidade direita da FEI, partindo $\lambda \to +\infty$, e vai diminuindo o valor do mesmo, caminhando da direita para a esquerda, até encontrarmos a extremidade esquerda da FEI (λ=0). Dessa forma, o primeiro passo é encontrar a curva que está por baixo no

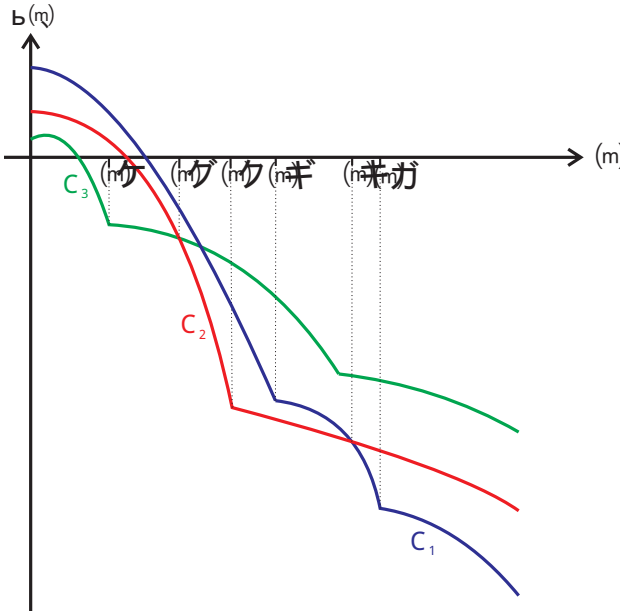


Figura 3.4: Pontos chave do exemplo anterior.

infinito, que nesse caso é aquela associada à carteira de maior retorno, ou seja, C_1.

De posse dessa informação, partindo de $\lambda \to +\infty$, caminhamos na curva que está por baixo até encontrarmos o seu primeiro ponto de mudança de segmento de parábola, que ocorre na coordenada λ_1. Para percorrer a curva C_1, foi necessário **um algoritmo, baseado no método de restrições ativas, que determina, de forma exata, cada segmento de parábola, associando-o a um conjunto ativo ótimo**. Nosso próximo passo é testar se, de fato, C_1 continua sendo a curva ótima em λ_1, o que realmente ocorre. Para realizar esse feito, foi necessário **um algoritmo de programação quadrática inteira mista, que encontra a solução de (3.1) para valores fixos de $\boldsymbol{\lambda}$**. Na análise feita nesse exemplo, vamos desconsiderar a hipótese de dois segmentos de parábola se interceptarem duas vezes, de forma a podermos afirmar que a curva que está por baixo no intervalo $[\lambda_1,+\infty)$ é C_1.

Partindo agora de λ_1, continuamos a caminhar na curva ótima, C_1, até encontrarmos um novo ponto de mudança de segmento de parábola, que ocorre na coordenada λ_3. Novamente, testamos se a curva que estamos percorrendo é ótima nessa coordenada. Nesse caso, a curva que está por baixo é C_2 de modo que existe uma coordenada, $\lambda_2 \in [\lambda_3,\lambda_1]$, em que as duas curvas se encontram. Para encontrá-la, precisamos caminhar da esquerda para a direita, partindo de λ_3, sobre a curva C_2, até encontrarmos o ponto em que ela cruza a curva C_1. Para isso, precisamos tanto de **um algoritmo que percorra essa curva no sentido contrário de nossa orientação**, quanto de **um algoritmo que encontre a interseção de duas curvas compostas**

por carteiras distintas. Como é possível que a curva ótima no ponto de interseção seja outra, também precisamos testar se a curva que está por baixo em λ_2 é, de fato, C_1 ou C_2. Em nosso exemplo, isso realmente ocorre, pois C_3 está acima das duas nesse ponto. Logo, podemos afirmar que a curva que está por baixo em $[\lambda_2,+\infty)$ é C_1 e a curva ótima em $[\lambda_3,\lambda_2]$ é C_2.

Agora, partindo da coordenada λ do ponto mais à esquerda conhecido da FEI, ou seja, λ_3, caminhamos na curva C_2 até seu próximo ponto de mudança de segmento de parábola, que ocorre na abscissa λ_4. Como a curva que está por baixo nessa coordenada permanece a mesma, ou seja, C_1 e C_3 continuam acima dela, podemos afirmar que C_2 é a curva ótima em $[\lambda_4,\lambda_2]$.

Seguindo nosso raciocínio, partindo de λ_4, caminhamos novamente sobre a curva C_2 até encontramos seu próximo ponto mudança de segmento de parábola, que ocorre quando $\lambda = 0$. Mais uma vez, testamos se a curva ótima nessa coordenada muda. Nesse caso, a curva que está por baixo é C_3, de modo que precisamos encontrar a coordenada de interseção das duas curvas, λ_5, por meio do mesmo processo que foi utilizado para determinar o encontro das duas primeiras curvas. Depois de encontrada a interseção, testamos se existe alguma curva abaixo do intercepto. Como nesse ponto, C_1 está acima das outras duas curvas, podemos dizer que a curva ótima em $[\lambda_5,\lambda_2]$ é C_2 e a curva que está por baixo em $[0,\lambda_5]$ é C_3.

Vale notar que a curva C_3 possui um ponto de mudança de segmento parábola em λ_6, que, apesar de não influir no intervalo em que essa curva é ótima, deve ser descoberto pelo nosso algoritmo, pois esses pontos, quando estão na curva que está por baixo, têm um papel fundamental na determinação da FEI do Problema (3.1)

A partir da análise desse exemplo, fica claro que, para executar a nossa proposta, é necessário combinar um certo número de algoritmos distintos. A seguir, vamos apresentar, de forma breve, os três principais algoritmos que foram combinados em nosso programa principal, que será chamado de *Algoritmo PFV*.

1. **Um método que, fixada a curva com que estamos trabalhando, percorra cada parábola que a compõe.** Esse método, que pode caminhar tanto da direita para a esquerda (diminuindo o valor de λ), quanto da esquerda para a direita (aumentando o valor de λ), nos permite percorrer a curva que está por baixo. Chamaremos-o de *Algoritmo Segmento FEI* .

2. **Um método para determinar se um ponto de uma curva fixada está por baixo das demais.** Essa verificação pode ser feita de forma direta ao resolvermos o Problema (3.1) com λ fixado, pois as variáveis binárias z_i da solução discriminam qual é a carteira ótima nesse ponto. Esse método é útil para determinarmos se um ponto final de um segmento de parábola de uma curva fixada está por baixo das demais curvas. O método será chamado de *Algoritmo PQIM*.

3. **Um algoritmo para encontrar a interseção de segmentos de parábola que estão em curvas compostas por carteiras distintas.** Esse método é necessário para corrigirmos o caminho a ser trabalhado, caso o Algoritmo PQIM determine que o ponto que estamos investigando não está na curva mais abaixo. Esse método será chamado de *Algoritmo das Interseções*.

O Algoritmo Segmento FEI será discutido em detalhes no Capítulo 4, ao passo que o Algoritmo PQIM será trabalhado no Capítulo 5 e o Algoritmo das Interseções será visto no Capítulo 6. Ainda no Capítulo 6, descreveremos como os três algoritmos foram combinados, bem como a proposta para se encontrar a curva que está por baixo.

Capítulo 4

A fronteira eficiente do problema com variáveis contínuas

Neste capítulo, vamos apresentar o método que usamos para encontrar a fronteira eficiente de problemas de portfólio em que as variáveis são reais, ou seja, os ativos que vão compor a carteira são conhecidos. Problemas desse tipo, em geral, podem ter sua fronteira eficiente determinada por algoritmos de Programação Quadrática Paramétrica (PQP), método bastante utilizado na literatura. Entre os métodos desse tipo estudados, destacamos o trabalhos de Best [5] e Stein [35], que se baseiam em um algoritmo de restrições ativas. Dentre esses, optamos por trabalhar com a proposta de Stein [35], devido ao fato do método ser numericamente estável para problemas de grande porte.

Conforme discutido no capítulo anterior, fixado o valor das variáveis inteiras, nosso algoritmo anda individualmente em cada parábola que a compõe, ou seja, ele caminha sobre a fronteira eficiente de Subproblemas (3.2). Dessa forma, baseados nas ideias de Stein [35], vamos propor um algoritmo que aproxima por completo a fronteira eficiente do Subproblema (3.2), assim como o Algoritmo Segmento FEI .

4.1 O problema geral

O algoritmo de Programação Quadrática Paramétrica com o qual trabalhamos nesse capítulo resolve o seguinte problema para todo $\lambda \in [0, +\infty)$:

$$\begin{array}{rl} \min & \frac{1}{2}x^T Q x - \lambda \mu^T x \\ S.a & A_I x \leq b_I \\ & A_E x = b_E, \end{array} \tag{4.1}$$

onde x,$\mu \in \mathbb{R}^n, Q \in \mathbb{R}^{nxn}$ é uma matriz simétrica e definida positiva, $A_I \in \mathbb{R}^{m_I \times n}$ e $A_E \in \mathbb{R}^{m_E \times n}$ são as matrizes das restrições de desigualdade e igualdade, com os seus respectivos vetores do lado direito $b_I \in \mathbb{R}^{m_I}$ e $b_E \in \mathbb{R}^{m_E}$.

Podemos encarar (4.1) como um problema de investimento com restrições lineares. Vale notar que, uma vez que a matriz de covariância é simétrica e definida positiva, o Subproblema (3.2) é um caso particular do Problema (4.1) em que não há as restrições de igualdade e os termos da restrição de desigualdade são da seguintes forma:

$$A_I = \begin{bmatrix} e^T \\ -I \end{bmatrix} \quad \text{e} \quad b_I = \begin{bmatrix} 1 \\ -l \end{bmatrix},$$

onde $e = (1,1,1,...,1)^T$, I é a matriz identidade de ordem n e l é o vetor de limite inferior das variáveis.

Na próxima seção, será apresentado o embasamento teórico do método.

4.2 Um método de restrições ativas

Nesta seção, vamos descrever um algoritmo de restrições ativas para encontrar uma aproximação da FEI de problemas do tipo (4.1). O algoritmo baseia-se na separação das inequações do problema em ativas e inativas, quando aplicadas a um dado ponto $\hat{x}$. Uma restrição linear do tipo $a_i^T x \leq b_i$ é ativa em $\hat{x}$ se $a_i^T \hat{x} = b_i$. Caso contrário ($a_i^T \hat{x} < b_i$), a restrição é dita inativa. Seguindo esse raciocínio, equações podem ser interpretadas como restrições sempre ativas. O conjunto de todas as restrições ativas é chamado de conjunto ativo. Chamaremos de I_a o conjunto dos índices das restrições ativas. A fronteira eficiente do nosso problema contínuo pode ser caracterizada por segmentos adjacentes, cada qual caracterizado por um conjunto ativo I_a^k constante.

Para iniciar o algoritmo, é necessário obter a solução em uma das extremidades da fronteira eficiente, seja ela a inicial ($\lambda = 0$) ou a final ($\lambda \to +\infty$). Caso se opte pela extremidade inicial,

o termo linear da função objetivo será nulo e teremos um problema de programação quadrática, ao passo que, se optarmos pela extremidade final, o termo linear será muito mais importante do que o quadrático, de forma que esse último possa ser desprezado, levando assim a um problema de programação linear. Como, em geral, é mais barato resolver um problema de programação linear, optamos por iniciar o algoritmo pela extremidade final da fronteira eficiente. Dessa forma, o problema de programação linear a ser resolvido tem a seguinte forma:

$$\begin{array}{rl} \min & -\mu^T x \\ S.a & A_I x \leq b_I \\ & A_E x = b_E. \end{array} \tag{4.2}$$

Caso a solução do Problema (4.2) seja única, ela estará no final da fronteira eficiente. Se a solução não for única, ou seja, temos soluções alternativas, encontramos a solução de termo quadrático mínimo, resolvendo o seguinte problema:

$$\begin{array}{rl} \min & \frac{1}{2} x^T Q x \\ S.a & \mu^T x = \rho^* \\ & A_I x \leq b_I \\ & A_E x = b_E, \end{array}$$

onde ρ^* é o valor ótimo da função objetivo de (4.2).

No caso dos problemas de investimento, esse processo é equivalente a encontrar a solução de menor risco dentre aquelas que possuem retorno máximo. Esse cenário aparece quando existem dois ativos distintos que possuem o retorno esperado máximo.

Uma vez que é mais fácil encontrar a solução inicial na extremidade direita da fronteira, vamos procurar os segmentos da fronteira eficiente caminhando da direita para a esquerda, de forma que cada segmento está associado a um conjunto I_a fixado. Esse processo é equivalente a, em um segmento qualquer, iniciarmos do maior valor possível de λ, diminuindo-o até encontrar uma quebra de segmento, ou seja, uma mudança no conjunto ativo. Após determinarmos a quebra, guardamos o valor de λ, o novo conjunto ativo atrelado ao próximo segmento a ser investigado e as informações do ponto correspondente ao final do segmento, para assim reaplicarmos o algoritmo utilizando o ponto encontrado como o ponto de partida. Tal algoritmo é aplicado até encontramos a solução na extremidade esquerda da fronteira eficiente do problema.

Em uma iteração qualquer do método, trabalha-se a partir do ponto mais à direita de um intervalo. Chamando de λ_1 a abscissa desse ponto, e de x^1 a solução do Problema (4.1) com esse valor de λ fixado, é necessário descobrir para que valores de λ o conjunto ativo ainda é ótimo. Em

outras palavras, deve-se encontrar o valor λ_2 em que ocorre uma mudança de conjunto ativo ótimo, assim como a solução correspondente, x^2. Para isso, devemos resolver um problema do tipo (4.1) em que o índice das restrições ativas é o mesmo da solução que corresponde ao início do segmento. A Figura 4.1 exemplifica a situação descrita.

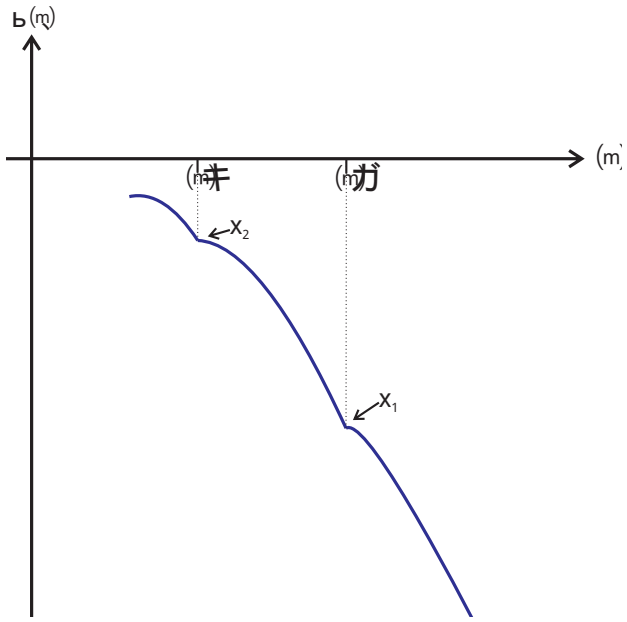


Figura 4.1: Segmento da fronteira eficiente do Problema (4.1).

Podemos garantir que o Problema (4.1) tem solução global ótima para um valor fixo de λ, pois exigimos que Q seja simétrica definida positiva, o que faz com que as condições de segunda ordem do problema sejam sempre satisfeitas e a solução do sistema KKT seja ótima. O menor valor de λ para qual o conjunto ativo fixado ainda é ótimo, ou seja, aquele em que o sistema KKT muda, corresponde ao ponto de quebra do segmento. Para encontrarmos os valores de x^* e λ^* associados a essa quebra, é necessário montar o sistema KKT associado às restrições ativas nesse segmento,

Nas condições KKT relacionadas às restrições ativas, o gradiente da função objetivo deve ser uma combinação linear do gradiente dessas restrições. Além disso, as restrições ativas devem ser satisfeitas na igualdade. Chamando de A_a e b_a a matriz e o vetor do lado direito relacionados às restrições ativas de um conjunto I_a qualquer e chamando de y o vetor de multiplicadores de Lagrange das restrições ativas do problema, notamos que $\nabla f(x) = Qx - \lambda\mu$ e o gradiente das restrições ativas é A_a. Portanto, denotando por y_I os multiplicadores de Lagrange das restrições

de desigualdade ativas, temos o seguinte sistema para as restrições que estão ativas:

$$\begin{cases} Qx - \lambda\mu + A_a^T y = 0 \\ A_a x = b_a \\ y_I \geq 0, \end{cases}$$

cujas equações, em notação matricial, podem ser reescritas como

$$\begin{bmatrix} Q & A_a^T \\ A_a & 0 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b_a \end{pmatrix} + \lambda \begin{pmatrix} \mu \\ 0 \end{pmatrix}. \tag{4.3}$$

Para determinar o final do segmento relativo ao conjunto ativo I_a, devemos resolver (4.3) e determinar qual será o primeiro multiplicador de Lagrange, y, a zerar, ou qual será a primeira restrição inativa que vai se tornar ativa.

Depois de resolvermos o sistema, devemos determinar o novo conjunto ativo, verificando quais são as restrições inativas que passam a ser ativas ou vice versa. Chamando de

$$H = \begin{bmatrix} Q & A_a^T \\ A_a & 0 \end{bmatrix}$$

a matriz do sistema, podemos reescrevê-lo como

$$H \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b_a \end{pmatrix} + \lambda \begin{pmatrix} \mu \\ 0 \end{pmatrix}. \tag{4.4}$$

Como a matriz H desse sistema pode ou não ser singular, devemos considerar duas abordagens diferentes para resolver esse sistema, uma para cada caso. Vamos explicá-las nas subseções a seguir.

4.2.1 Caso H seja não singular

Caso H seja não singular, a solução do sistema (4.4) é única, o que nos permite particionar (4.4) em dois sistemas lineares mais simples:

$$H \begin{bmatrix} h_{x,q} \\ h_{y,q} \end{bmatrix} = \begin{pmatrix} 0 \\ b_a \end{pmatrix} \quad \text{e} \quad H \begin{bmatrix} h_{x,p} \\ h_{y,p} \end{bmatrix} = \begin{pmatrix} \mu \\ 0 \end{pmatrix}, \tag{4.5}$$

sendo

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{pmatrix} h_{x,q} + \lambda^* h_{x,p} \\ h_{y,q} + \lambda^* h_{y,p} \end{pmatrix},$$

onde λ^* é o valor de λ no final do segmento atual.

Depois de obtidas as soluções dos dois sistemas, precisamos determinar qual o valor de λ^*, assim como o novo conjunto ativo. Para isso, precisamos determinar para qual valor de λ o segmento acaba, o que ocorre se uma das restrições que eram inativas se torna ativa ou se uma restrição que era ativa se torna inativa. Para encontrar o valor de λ^* em que a primeira restrição inativa se torna ativa, basta notar que, para as restrições inativas,

$$Ax \leq b \rightarrow A(h_{x,q} + \lambda^* h_{x,p}) \leq b \rightarrow \lambda^* \geq \frac{b_i - a_i^T h_{x,q}}{a_i^T h_{x,p}}, \text{ para } i \notin I_a \text{ e } a_i^T h_{x,p} < 0.$$

Como precisamos determinar o menor valor de λ que faz com que as inequações referentes às restrições inativas sejam satisfeitas de forma estrita, e estamos caminhando da direita para a esquerda, diminuindo o valor de λ, devemos adotar:

$$\hat{\lambda} = max \left\{ \frac{b_i - a_i^T h_{x,q}}{a_i^T h_{x,p}}, \text{ para } i \notin I_a \text{ e } a_i^T h_{x,p} < 0 \right\}. \tag{4.6}$$

Por outro lado, uma restrição ativa se torna inativa se o multiplicador de Lagrange associado se torna zero na solução do sistema KKT. Uma vez que os multiplicadores de Lagrange associados às restrições de desigualdade do problema possuem o sinal usual, ou seja, são sempre positivos, podemos impor que:

$$h_{y,q,i} + \tilde{\lambda} h_{y,p,i} \geq 0 \rightarrow \tilde{\lambda} \geq \frac{-h_{y,q,i}}{h_{y,p,i}}, \text{ para } h_{y,p,i} > 0 \text{ e } i \in I_a.$$

Para encontrarmos o primeiro multiplicador de Lagrange que vai se tornar zero, devemos adotar:

$$\tilde{\lambda} = max \left\{ \frac{-h_{y,q,i}}{h_{y,p,i}}, \text{ para } i \in I_a \text{ e } h_{y,p,i} > 0 \right\}. \tag{4.7}$$

Para achar qual das possibilidades ocorre primeiro, adotamos $\lambda = max\{\hat{\lambda},\tilde{\lambda}\}$. Assim, no final de uma iteração qualquer k, há duas formas de atualizar o conjunto ativo I_a^k. Se λ for determinado por (4.6) e o índice da restrição que entra é r, fazemos $I_a^{k+1} = I_a^k \cup r$. Se λ for encontrado por (4.7) e o índice da restrição que sai é s, temos $I_a^{k+1} = I_a^k \setminus s$.

4.2.2 Caso H seja singular

A matriz H nem sempre é não singular, ou seja, o sistema (4.4) não necessariamente tem solução única. Essa situação ocorre com frequência quando estamos procurando a solução inicial

do problema. Nesse caso, é comum termos que resolver um problema tal como

$$\begin{aligned} \min \quad & -\mu^T x \\ S.a \quad & x^T e = 1 \\ & x \leq 0{,}1 \\ & x \geq 0. \end{aligned}$$

Uma vez que esse problema é equivalente ao qual se maximiza $\mu^T x$, e como o limitante superior de cada variável é 0,1, a melhor solução consiste em atribuirmos o valor 0,1 para as dez variáveis cujo valor μ_i associado seja o maior possível. Dessa forma, teremos dez variáveis no limitante superior e o resto no limitante inferior fazendo com que todos os vetores canônicos (colunas da matriz identidade) estejam em A_a. Além disso, a primeira restrição também é ativa, por ser de igualdade, fazendo com que o vetor e também esteja em A_a. Como e pode ser escrito como uma combinação linear das colunas da matriz identidade, A_a não possui posto completo.

A dificuldade discutida acima também pode ocorrer em uma iteração qualquer do método, principalmente se a solução inicial não possuir posto completo. Uma maneira de se contornar tal problema é minimizar o valor de λ dentre todas as soluções possíveis do sistema KKT. Para isso, resolvemos o seguinte problema de programação linear:

$$\begin{aligned} \min \quad & \lambda_{n+1} \\ S.a \quad & H \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b_a \end{pmatrix} + \lambda_{n+1} \begin{pmatrix} \mu \\ 0 \end{pmatrix} \\ & Ax \leq b \\ & y_I \geq 0. \end{aligned} \tag{4.8}$$

Os elementos de x determinam a solução no final do segmento. Já os elementos de y que são 0 indicam quais restrições ativas se tornarão inativas no próximo segmento. Para descobrirmos quais restrições inativas vão se tornar ativas, basta usar o vetor x encontrado para calcular as inequações $Ax \leq b$ do problema original e ver quais restrições passam a ser de igualdade. Por construção, tal procedimento garante otimalidade no final do segmento, sendo $[\lambda_{n+1},\lambda_n]$ o intervalo correspondente ao segmento obtido.

4.3 Um algoritmo para obter a fronteira eficiente do problema com variáveis reais

Por meio da teoria apresentada na seção anterior, implementamos o método proposto por Stein [35], que obtém a FEI do Problema (4.1) e será chamado ao longo do texto de *Algoritmo FEI*. Seu pseudocódigo é ilustrado no Algoritmo 1.

Algoritmo 1: O Algoritmo FEI

1 Calcule a solução inicial x^1;
2 Determine o conjunto de restrições ativas associado, I_a^1;
3 Faça $k \leftarrow 1$ e $\lambda_k \leftarrow +\infty$;
4 Calcule a matriz H, assim como a sua decomposição LU;
5 **se** *a decomposição LU for única, ou seja, se H for não singular* **então**
6 | Calcule os vetores $h_p = \begin{bmatrix} h_{x,q} \\ h_{y,q} \end{bmatrix}$ e $h_q = \begin{bmatrix} h_{x,p} \\ h_{y,p} \end{bmatrix}$;
7 | Calcule $\hat{\lambda}$, $\tilde{\lambda}$ e faça $\lambda_{k+1} \leftarrow max\{\hat{\lambda},\tilde{\lambda}\}$ de acordo com (4.6) e (4.7);
8 | Faça $x^{k+1} \leftarrow h_{x,q} + \lambda_{k+1} h_{x,p}$;
9 **senão**
10 | Encontre λ_{k+1} e x^{k+1} resolvendo (4.8);
11 **fim**
12 Determine o novo conjunto ativo, I_a^{k+1};
13 **se** $\lambda_{k+1} \geq 10^{-6}$, **então**
14 | Guarde λ_{k+1}, x^{k+1};
15 | Faça $k \leftarrow k+1$ e volte para a linha 4;
16 **senão**
17 | Faça $x^{k+1} \leftarrow \frac{\lambda_k}{\lambda_k - \lambda_{k+1}} x^{k+1} + \frac{\lambda_{k+1}}{\lambda_k - \lambda_{k+1}} x^k$ e $\lambda_{k+1} \leftarrow 0$;
18 **fim**

Nesse algoritmo, primeiramente calculamos a solução de (4.1) na extremidade direita ($\lambda \rightarrow +\infty$), assim como o conjunto ativo fixo correspondente (linhas 1 e 2). Logo após, calculamos a matriz H e determinamos se ela é singular ou não por meio de sua decomposição LU (linha 4). Caso ela não seja, determinamos o valor de λ correspondente à mudança de conjunto ativo, assim como a solução de (4.1) no final do segmento (linhas 6 a 8). Caso contrário, essa determinação é feita por meio de (4.8) (linha 10). Em seguida, calculamos o novo conjunto ativo (linha 12) e testamos se ultrapassamos a extremidade esquerda da FEI (linha 13). Caso isso não ocorra, ou seja, se λ_{k+1} é positivo, atualizamos o contador e voltamos a linha 4 (linhas 14 e 15), partindo agora do ponto mais a esquerda conhecido da FEI. Senão, o conjunto ativo atual é ótimo na extremidade esquerda da FEI, de forma que podemos terminar o algoritmo ao atualizar os valores de x e λ

(linha 17).

A fronteira eficiente do problema é formada pelos segmentos adjacentes que ligam os pontos de mudança de valor de λ associados à troca de conjunto ativo. Para tornar o algoritmo mais eficiente, na prática, resolve-se os sistemas de (4.5) por meio da decomposição LU da matriz H.

No Capítulo 6, vamos mostrar como determinar graficamente a FEI de nosso problema a partir dos segmentos obtidos nesse algoritmo, assim como uma maneira de aproximar a solução de (3.2) para qualquer valor de λ.

4.3.1 Um exemplo

Consideremos o Problema (4.1), com $n = 3$ e sem restrições de igualdade. Tomemos como dados iniciais:

$$Q = \begin{bmatrix} 0{,}1876 & 0{,}2910 & 0{,}0685 \\ 0{,}2910 & 1{,}2346 & 0{,}5953 \\ 0{,}0685 & 0{,}5953 & 0{,}4270 \end{bmatrix}, \mu = \begin{bmatrix} 0{,}1343 \\ 0{,}0986 \\ 0{,}1420 \end{bmatrix}, A_I = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, b_I = \begin{bmatrix} 1 \\ -0{,}2 \\ -0{,}2 \\ -0{,}2 \end{bmatrix}.$$

Aplicando o algoritmo apresentado nessa seção ao problema, temos:

- Linha 1: A solução inicial é $x^1 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}6 \end{bmatrix}$.
- Linha 2: As restrições 1,2 e 3 estão ativas, logo $I_a^1 = \{1,2,3\}$.

A parte iterativa do método é apresentada abaixo:

- Iteração 1.
  - Linha 5 H não é singular, então vamos para a linha 7.
  - Linha 7: Os valores de λ obtidos são: $\hat{\lambda} = -\infty$, $\tilde{\lambda} = 32{,}7455$ e $\lambda_2 = 32{,}7455$.
  - Linha 8: A solução é $x^2 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}6 \end{bmatrix}$.
  - Linha 12: Como $\lambda_2 = \tilde{\lambda} = 32{,}7455$, uma restrição ativa se torna inativa. Nesse caso, trata-se da restrição 2, logo $I_a^2 = \{1,3\}$.

- Linha 13: $\lambda_2 > 0$, de modo que voltamos a linha 4.

- Iteração 2.
  - Linha 5: H não é singular, então vamos para a linha 7.
  - Linha 7: Os valores de λ obtidos são: $\hat{\lambda} = 7{,}9351$, $\tilde{\lambda} = 1{,}4660$ e $\lambda_3 = 7{,}9351$.
  - Linha 8: A solução é $x^3 = \begin{bmatrix} 0{,}6 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.
  - Linha 12: Como $\lambda_3 = \hat{\lambda} = 7{,}9351$, uma restrição inativa se torna ativa. Nesse caso, a restrição 4, logo $I_a^3 = \{1{,}3{,}4\}$.
  - Linha 13: $\lambda_3 > 0$, de modo que voltamos a linha 4.

- Iteração 3.
  - Linha 5 H não é singular, então vamos para a linha 7.
  - Linha 7: Os valores de λ obtidos são: $\hat{\lambda} = -\infty$, $\tilde{\lambda} = 1{,}3735$ e $\lambda_4 = 1{,}3735$.
  - Linha 8: A solução é $x^4 = \begin{bmatrix} 0{,}5999 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.
  - Linha 12: Como $\lambda_4 = \tilde{\lambda} = 1{,}3735$, uma restrição ativa se torna inativa. Nesse caso, a restrição 1, logo $I_a^4 = \{3{,}4\}$.
  - Linha 13: $\lambda_4 > 0$, de modo que voltamos a linha 4.

- Iteração 4.
  - Linha 5: H não é singular, então vamos para a linha 7
  - Linha 7: Os valores de λ obtidos são: $\hat{\lambda} = 0{,}8147$, $\tilde{\lambda} = -2{,}3190$ e $\lambda_5 = 0{,}8147$.
  - Linha 8: A solução é $x^5 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.
  - Linha 12: Como $\lambda_5 = \hat{\lambda} = 0{,}8147$, uma restrição inativa se torna ativa. Nesse caso, a restrição 2, logo $I_a^5 = \{2{,}3{,}4\}$.
  - Linha 13 $\lambda_5 > 0$, de modo que voltamos a linha 4.

- Iteração 5.
  - Linha 5: H não é singular, então vamos para a linha 7
  - Linha 7: Os valores de λ obtidos são: $\hat{\lambda} = -\infty$, $\tilde{\lambda} = -\infty$ e $\lambda_6 = -\infty$.
  - Linha 8: Passo não aplicável, pois $\lambda_6 = -\infty$.
  - Linha 12: O conjunto ativo não muda, logo $I_a^6 = \{2,3,4\}$.
  - Linha 13: Nesse caso, como $\lambda_6 < 0$, vamos para a a linha 17
  - Linha 17: $\lambda_6 = 0$ e $x^6 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.
- Linha 18: Fim

A análise desse exemplo revela que a fronteira eficiente do problema é composta pelos seguintes segmentos:

1. Um segmento em que $\lambda \in [32{,}7455; +\infty)$, o conjunto ativo é $I_a^1 = \{1,2,3\}$ e x varia entre $x^1 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}6 \end{bmatrix}$ e $x^2 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}6 \end{bmatrix}$.
2. Um segmento em que $\lambda \in [7{,}9351; 32{,}7455]$, o conjunto ativo é $I_a^2 = \{1,3\}$ e x varia entre $x^2 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}6 \end{bmatrix}$ e $x^3 = \begin{bmatrix} 0{,}6 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.
3. Um segmento em que $\lambda \in [1{,}3735; 7{,}9351]$, o conjunto ativo é $I_a^3 = \{1,3,4\}$ e x varia entre $x^3 = \begin{bmatrix} 0{,}6 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$ e $x^4 = \begin{bmatrix} 0{,}5999 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.
4. Um segmento em que $\lambda \in [0{,}8147; 1{,}3735]$, o conjunto ativo é $I_a^4 = \{3,4\}$ e x varia entre $x^4 = \begin{bmatrix} 0{,}5999 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$ e $x^5 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.

5. Um segmento em que $\lambda \in [0; 0{,}8147]$, o conjunto ativo é $I_a^5 = \{2{,}3{,}4\}$ e x varia entre $x^5 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$ e $x^6 = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}$.

O segmento 1 corresponde ao intervalo em que apenas o retorno é priorizado, sendo a melhor solução investir o máximo possível no ativo mais rentável, mantendo o investimento mínimo nos demais.

O segmento 5 corresponde ao intervalo em que apenas o risco é priorizado, sendo a melhor solução investir o mínimo possível.

As Figuras 4.2 e 4.3 mostram a FEI desse exemplo obtida por meio do software MATLAB.

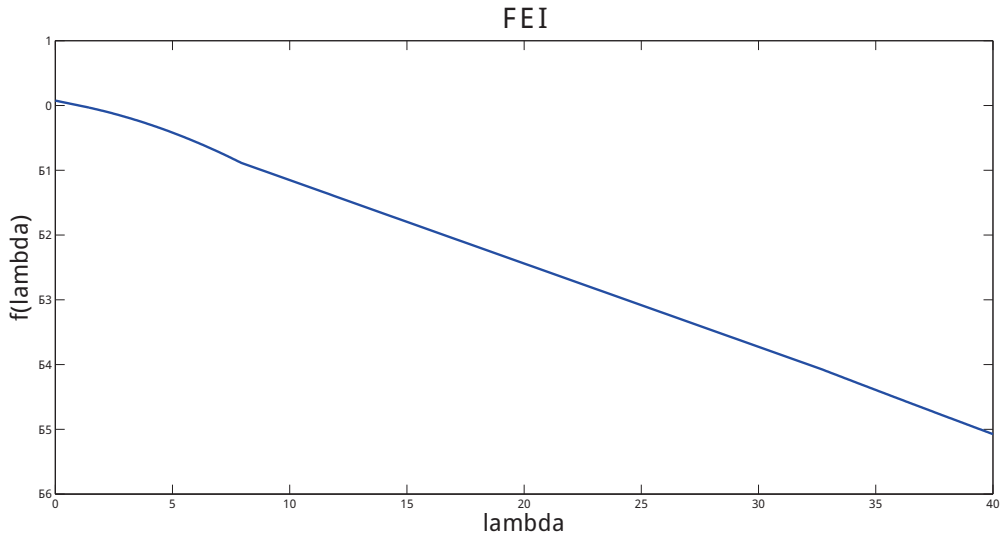


Figura 4.2: FEI para $\lambda \in [0{,}40]$.

4.3.2 O Algoritmo Segmento FEI

O algoritmo proposto nessa seção aproxima por completo a fronteira eficiente do Problema (4.1) e consequentemente a FEI do Problema (3.2). No Capítulo 3, vimos a necessidade de um algoritmo que caminhe sobre um segmento de parábola da fronteira eficiente do Problema (3.2) (Algoritmo Segmento FEI). Tal processo é equivalente a caminhar sobre um segmento da fronteira eficiente em que o conjunto ativo é constante, ou seja, um passo do algoritmo geral apresentado acima.

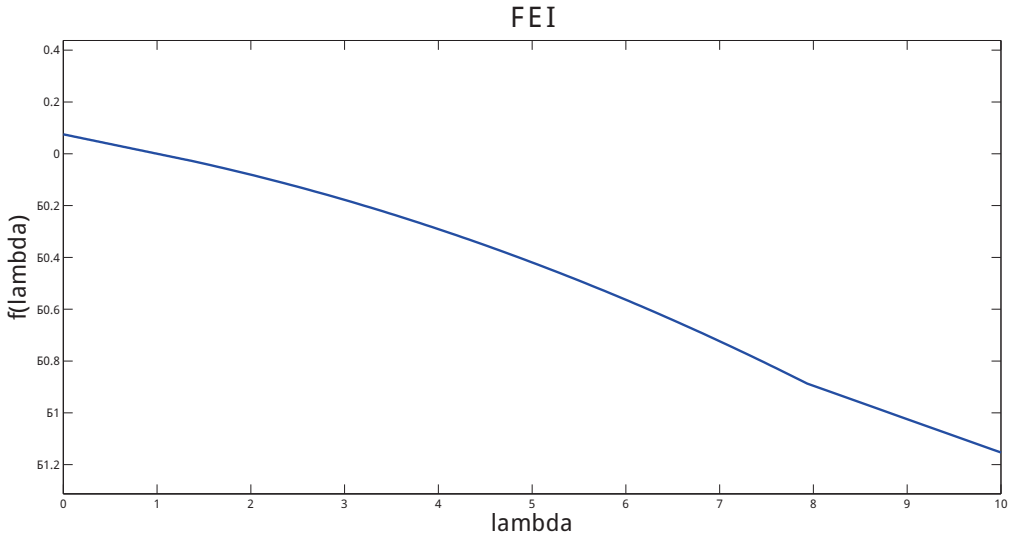


Figura 4.3: FEI para $\lambda \in [0,10]$.

Uma vez que o Algoritmo Segmento FEI precisa caminhar sobre a curva tanto da direita para a esquerda (reduzindo o valor de λ) quanto da esquerda para a direita (aumentando o valor de λ), podemos dividi-lo em duas partes, que chamaremos de Algoritmo Segmento FEI-DE e Algoritmo Segmento FEI-ED, cada uma referente a um sentido da trajetória. Como o algoritmo da seção anterior só caminha da direita para a esquerda, cada iteração do mesmo aplicado ao Problema (3.2) é exatamente o Algoritmo Segmento FEI-DE. Esse algoritmo, que parte do valor de λ_1 associado à extremidade direita do segmento da FEI que estamos procurando, nada mais é do que uma iteração do algoritmo apresentado nessa seção.

Nesse caso, o segmento da FEI procurado é aquele em que o conjunto ativo I_a^1 está fixado e $\lambda \in [\lambda_2,\lambda_1]$. Nesse caso, x^2 é a solução de (3.2) com $\lambda = \lambda_2$ fixado e x^1 é a solução de (3.2) com $\lambda = \lambda_1$ fixado. Essas informações são suficientes para descrever o segmento de parábola correspondente a esse intervalo, conforme veremos com mais detalhes no Capítulo 6.

4.4 Caminhando da esquerda para a direita

Conforme discutido anteriormente, é possível caminhar em um segmento da fronteira eficiente de (4.1) tanto da direita para a esquerda, quanto da esquerda para a direita. No segundo caso, conhecemos o ponto inicial mais à esquerda do segmento (x^2) e procuramos o ponto final mais à direita do mesmo (x^1). Esse processo é análogo ao apresentado anteriormente, diferindo apenas no modo como o valor de λ no final do segmento é calculado. Novamente, devemos trabalhar com

o sistema (4.4) e considerar o caso da matriz H ser ou não singular.

4.4.1 Caso H seja não singular

Como visto anteriormente, caso H seja não singular, podemos dividir o Sistema (4.4) na forma (4.5) e determinar para qual valor de λ^* o segmento acaba, ou seja, descobrir quando uma da restrição inativa se torna ativa ou quando uma restrição ativa se torna inativa. No primeiro caso, basta notar que, para as restrições inativas,

$$Ax \leq b \rightarrow A(h_{x,q} + \lambda^* h_{x,p}) \leq b \rightarrow \lambda^* \leq \frac{b_i - a_i^T h_{x,q}}{a_i^T h_{x,p}}, \text{ para } i \notin I_a \text{ e } a_i^T h_{x,p} > 0.$$

Como estamos caminhando da esquerda para a direita, aumentando o valor de λ, o primeiro valor desse parâmetro que faz com que as inequações referentes às restrições inativas deixem de ser satisfeitas de forma estrita é:

$$\hat{\lambda} = min \left\{ \frac{b_i - a_i^T h_{x,q}}{a_i^T h_{x,p}}, \text{ para } i \notin I_a \text{ e } a_i^T h_{x,p} > 0 \right\}. \tag{4.9}$$

No segundo caso, uma restrição ativa se torna inativa se o multiplicador de Lagrange associado se torna zero na solução do sistema KKT. Escrevendo, então, o multiplicador de Lagrange do problema de forma análoga àquela apresentada na Seção 4.2.1, temos

$$h_{y,q,i} + \tilde{\lambda} h_{y,p,i} \geq 0 \rightarrow \tilde{\lambda} \leq \frac{-h_{y,q,i}}{h_{y,p,i}}, \text{para } h_{y,p,i} < 0 \text{ e } i \in I_a.$$

Para encontrarmos o primeiro multiplicador de Lagrange nulo, devemos adotar:

$$\hat{\lambda} = min \left\{ \frac{-h_{y,q,i}}{h_{y,p,i}}, \text{ para } i \in I_a \text{ e } h_{y,p,i} < 0 \right\}. \tag{4.10}$$

Para saber qual das possibilidades ocorre primeiro, adotamos $\lambda = min\{\hat{\lambda},\tilde{\lambda}\}$. Se λ for determinado por (4.9) e o índice da variável que entra for r, temos $I_a^{n+1} = I_a^n \cup r$. Se λ for encontrado por (4.10) e o índice da variável que sai for s, fazemos $I_a^{n+1} = I_a^n \setminus s$.

4.4.2 Caso H seja singular

Assim como foi visto na Subseção 4.2.2, caso H seja singular, a solução do sistema KKT não é única. Nesse caso, encontramos a solução que possui o maior valor possível de λ, resolvendo a versão de maximização do Problema (4.8).

A determinação do novo conjunto ativo é feita da mesma maneira apresentada na Subseção 4.2.2.

4.4.3 O Algoritmo Segmento FEI-ED

Utilizando a teoria dessa seção, implementamos o Algoritmo Segmento FEI-ED, que caminha da esquerda para a direita. Seu pseudocódigo é ilustrado no Algoritmo 2.

Algoritmo 2: O Algoritmo Segmento FEI-ED

```
1  Calcule a solução x^2 associada ao valor λ_2;
2  Determine o conjunto de restrições ativas associado I_a;
3  Calcule a matriz H, assim como a sua decomposição LU;
4  se a decomposição LU for única, ou seja, se H for não singular, então
5  |  Calcule os vetores [h_{x,q}; h_{y,q}] e [h_{x,p}; h_{y,p}];
6  |  Calcule λ̂, λ̃ e λ_1 ← min{λ̂,λ̃} de acordo com (4.9) e (4.10);
7  |  Faça x^1 ← h_{x,q} + λ_1 h_{x,p};
8  senão
9  |  Encontre λ_1 e x^1 resolvendo a versão de maximização do Problema (4.8);
10 fim
```

Esse algoritmo, que tem como entrada o valor de λ_2 associado à extremidade esquerda do segmento da FEI que estamos procurando e como saída o valor de λ_1 na extremidade direita do mesmo, é análogo ao Algoritmo Segmento FEI-DE, e é a primeira de nossas contribuições dentro desse trabalho. A única diferença aparece no modo como λ é calculado no final do segmento. Nesse caso, esse calculo é feito por meio de (4.9), (4.10) e da versão de maximização do Problema (4.8).

Capítulo 5

O problema com variáveis inteiras

Conforme foi discutido no Capítulo 3, o nosso algoritmo requer um método para determinar a FEI do Problema (3.1) com um determinado valor de λ, ou seja, resolver um problema da seguinte forma:

$$\begin{array}{rll} \min & f(x) = \frac{1}{2}x^T Q x + c^T x & \\ S.a & Ax \leq b & \\ & l_i z_i \leq x_i \leq z_i, & i = 1,...,n \\ & z_i \in \{0,1\}, & i = 1,...,n \\ & \sum_{i=1}^{n} z_i \leq K. & \end{array} \tag{5.1}$$

Esse é um problema de programação quadrática inteira mista 0-1, portanto NP-difícil [7]. Devido a essa complexidade e à necessidade de um método numericamente eficiente, decidimos explorar dois algoritmos diferentes: um que é barato, mas não necessariamente encontra a solução ótima, e outro que é mais caro, porém robusto. O mais barato é o *CHR* (*Convex Hull Relaxation*), que trabalha com a relaxação das restrições de integralidade sobre o envoltório convexo das mesmas, não garantindo que a solução ótima encontrada seja inteira. Já o mais robusto é o algoritmo de programação quadrática inteira mista, proposto por Bertsimas e Shioda [4] e aperfeiçoado por Villela [36], que combina o método de Lemke e o algoritmo *Branch-and-Bound* de forma implícita.

5.1 A relaxação no envoltório convexo

Uma primeira abordagem para resolver problemas de otimização não linear inteira, consiste em trabalhar com a relaxação das restrições de integralidade. Essa relaxação pode ser realizada de diferentes formas, dependendo das restrições que são relaxadas. Dentre elas, destacamos o caso da relaxação sobre o envoltório convexo (CHR), introduzida por Guignard e Ahlatçioglu [26], que foi

foco de nossos estudos.

Antes de apresentar o problema relaxado que o método CHR resolve, vamos definir matematicamente o conceito de envoltório convexo, que é a base para o seu entendimento.

O envoltório convexo de um conjunto $S \subset \mathbb{R}^n$, denotado por $C_0(S)$, é definido como o conjunto de todas as possíveis combinações convexas de elementos de S, ou seja:

$$C_0(S) = \left\{ \sum_{i=1}^{k} a_i x_i | x_i \in S, \sum_{i=1}^{k} a_i x_i = 1, a_i \geq 0 \right\}.$$

Claramente, $C_0(S)$ é um conjunto convexo que contém S, ou seja, $S \subset C_0(S)$. Além disso, é possível observar que o envoltório convexo é o menor conjunto convexo que contém S, de forma que, se R é um conjunto convexo tal que $S \subset R$, então $C_0(S) \subset R$. A figura abaixo resume essa ideia:

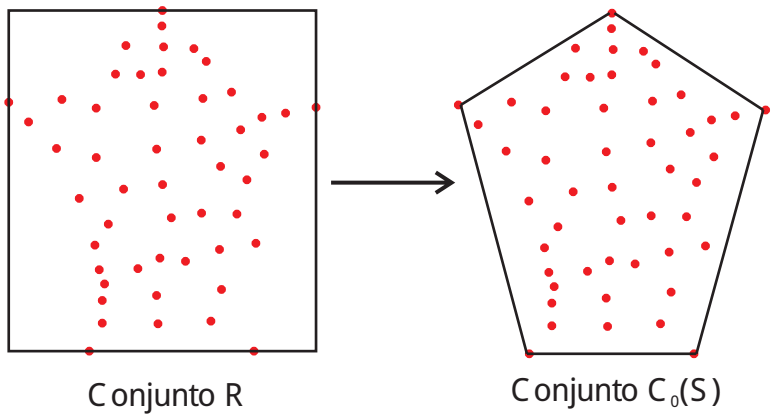


Figura 5.1: Envoltório convexo.

Para entendermos o método CHR, primeiramente devemos considerar o seguinte problema de otimização inteira:

$$\begin{aligned} \min \quad & f(x) \\ S.a \quad & x \in S, \end{aligned} \tag{5.2}$$

onde $f(x)$ é uma função não linear convexa de $x \in \mathbb{R}^n$, $S = \{x \in Y | Ax \leq b\}$, $A \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$ e Y é um subconjunto de $\mathbb{R}^n$ que especifica as restrições de integralidade sobre x.

Definimos a relaxação sobre o envoltório convexo do problema como:

$$\begin{aligned} \min \quad & f(x) \\ S.a \quad & x \in C_0(S), \end{aligned} \tag{5.3}$$

onde $C_0(S)$ é o envoltório convexo das restrições.

Note que, apesar de (5.3) ser uma relaxação primal, uma vez que $S \subset C_0(S)$, o CHR minimiza a função objetivo de (5.2) sobre o menor conjunto contínuo que contém S, evitando assim de relaxar as restrições lineares do problema original.

Para o método CHR convergir globalmente, as três condições abaixo devem ser satisfeitas:

1. A região factível deve ser compacta e convexa.

2. A função objetivo deve ser convexa.

3. As restrições devem ser lineares.

No caso do Problema (3.1), essas condições são verdadeiras, pois as restrições são lineares em um espaço de variáveis contínuas e binárias, além da função objetivo ser quadrática e convexa.

Como o método CHR resolve uma relaxação de um problema de programação não linear inteira, a solução obtida pelo mesmo pode ser fracionária. Entretanto, se a função objetivo $f(x)$ for linear, pode-se afirmar que um ponto ótimo de (5.3) será necessariamente um ponto extremo do envoltório convexo da região factível inteira original, ou seja, a solução obtida pelo CHR será inteira, coincidindo com a solução do Problema (5.2). A Figura 5.2 ilustra tal fato em um problema de programação linear inteira (PLI) de maximização que possui duas variáveis.

A maior dificuldade do problema que CHR resolve está na formulação implícita do envoltório convexo. Porém, é possível dividir (5.3) em um problema mestre e um subproblema linear por meio da ideia da decomposição simplicial, introduzida por Frank e Wolfe [25] em 1956. O algoritmo consiste em, a cada iteração, procurarmos, por meio do subproblema, um novo ponto extremo do envoltório convexo das restrições, ao passo que, por meio do problema mestre, encontramos a combinação linear dos pontos extremos já conhecidos que minimiza o valor de $f(x)$. Caso essa combinação linear esteja dentro de um simplex de uma dimensão menor, paramos o algoritmo, e a solução obtida pelo método CHR será a combinação encontrada pelo problema mestre. Caso contrário, voltamos ao subproblema utilizando essa solução como ponto de partida. Como a combinação linear ótima dos pontos extremos pode ser fracionária, nada garante que a solução obtida pelo algoritmo seja inteira. Nas subseções a seguir, vamos explicar em mais detalhes os dois problemas que o CHR resolve em cada iteração.

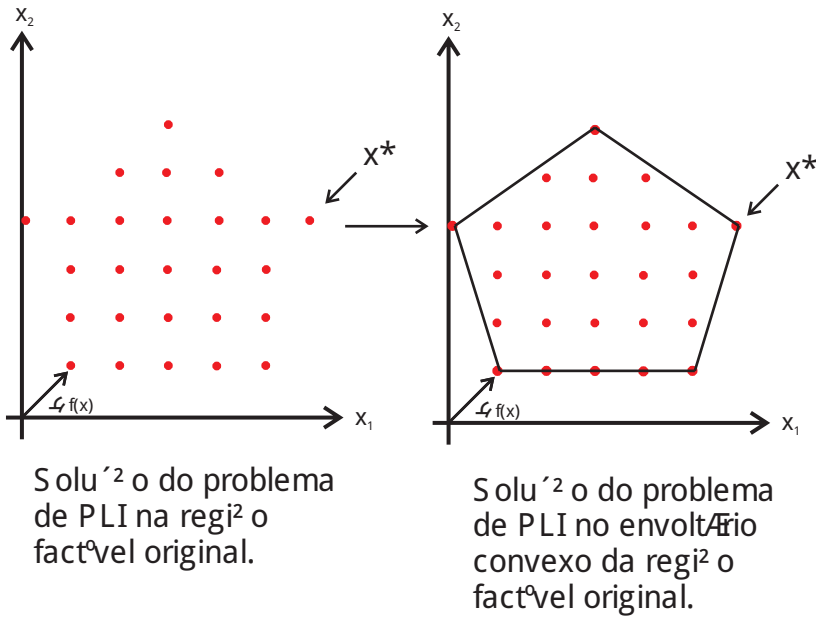


Figura 5.2: O CHR aplicado a um problema de programação linear inteira.

5.1.1 O subproblema de programação linear inteira mista

O subproblema a ser resolvido consiste na busca de uma direção de descida factível, que é muito frequente em problemas de programação não linear.

Na k-ésima iteração da nossa decomposição, dado um ponto factível de (5.3), x^k, precisamos encontrar a direção de descida, que faz o menor ângulo agudo com $\nabla f(x^k)$, no poliedro definido pelo envoltório convexo $C_0(S)$, ou seja, é necessário resolver o seguinte problema de programação linear para o iterando y^k:

$$\begin{array}{ll} \min & \nabla f(x^k)^T(y^k - x^k) \\ S.a & y^k \in C_0(S). \end{array} \tag{5.4}$$

Como a função objetivo é linear, o ótimo sobre o envoltório convexo dos pontos inteiros é necessariamente atingido em um ponto extremo de $C_0(S)$, ou seja, em um ponto inteiro da região factível do Problema (5.2). Utilizando esse fato, podemos transformar o Subproblema (5.4) em

um problema de programação linear inteira mista equivalente da seguinte forma:

$$\begin{aligned} \min \quad & \nabla f(x^k)^T(y^k - x^k) \\ S.a \quad & Ay^k \leq b \\ & y^k \in Y. \end{aligned} \tag{5.5}$$

Uma vez que o subproblema é linear, podemos achar o ótimo do mesmo por meio de vários métodos já existente para programação linear inteira mista.

A solução obtida do Subproblema (5.5), y^k, é um novo ponto do envoltório convexo das restrições, a não ser que x^k já seja ótimo para o (5.3). Nesse caso, prosseguimos para o problema mestre. De uma maneira geral, é fácil ver que, a cada iteração, aumentamos o número de pontos extremos do envoltório convexo conhecidos, de forma que, no pior dos casos, o método acaba na iteração em que todos esses eles passam a ser conhecidos.

5.1.2 O problema mestre

O problema mestre consiste em otimizar a função objetivo não linear inicial sobre o envoltório convexo dos pontos extremos gerados pelo Subproblema (5.5), o que torna o problema mais complexo a cada iteração. Suponha que, na iteração k, tenhamos r pontos extremos obtidos a partir dos Subproblemas (5.5) (note que, como a solução de (5.5) não é necessariamente única, podemos ter $r \neq k+1$). A partir deles, definimos uma matriz $C \in \mathbb{R}^{nxr}$, em que cada coluna C_i é um ponto extremo já obtido por (5.5), e um vetor w cujas coordenadas são os pesos da combinação convexa dos r pontos extremos. Dessa forma, o problema mestre é:

$$\begin{aligned} \min \quad & f(Cw) \\ S.a \quad & \sum_{i=1}^{r} w_i = 1 \\ & w_i \geq 0, i = 1,2,3,...,r. \end{aligned} \tag{5.6}$$

Note que, devido às restrições do problema, o produto Cw é a combinação convexa dos pontos extremos obtidos nos subproblemas anteriores. Logo, minimizamos o valor de $f(x)$ dentro do envoltório convexo gerado pelos pontos extremos, obtendo como solução os pesos da combinação convexa dos pontos extremos que minimiza o valor de $f(x)$. Sendo assim, tal problema é equivalente a minimizar f sobre um simplex de dimensão $r-1$.

Se a solução ótima obtida pelo método CHR estiver dentro desse simplex, então o algoritmo termina. Caso contrário, a combinação linear obtida com os pesos ótimo w^* é utilizada como o

próximo iterando, x^{k+1}, no Subproblema (5.5), ou seja, definimos

$$x^{k+1} = \sum_{i=1}^{r} w_i^* C_i \tag{5.7}$$

e voltamos ao subproblema, encontrando outro ponto extremo e aumentando a dimensão do simplex do problema mestre.

A forma mais prática de testar se a solução obtida pelo Subproblema (5.6) é ótima para (5.6) consiste em verificar se a solução do próximo Subproblema (5.5), y^{k+1}, gera uma direção de descida ortogonal ao $\nabla f(x^{k+1})$, ou seja, se $\nabla f(x^{k+1})^T(y^{k+1} - x^{k+1}) = 0$, sendo esse um dos critérios de parada do algoritmo.

5.1.3 O Algoritmo CHR

Apesar do Problema (5.3) possuir uma definição implícita, de difícil compreensão, a implementação de um método computacional para resolvê-lo é barata e intuitiva. Por meio das ideias apresentadas nessa seção, implementamos um algoritmo, que denotaremos por *Algoritmo CHR*, que encontra a solução desse tipo de problema Seu pseudocódigo é apresentado no Algoritmo 3.

Algoritmo 3: O Algoritmo CHR

```
1  Faça k ← 1 e ache um ponto extremo de (5.3), x^1;
2  Resolva o problema de programação linear inteira (5.5) usando o valor de x^k fixo, obtendo
   assim uma solução y^k;
3  se ||y^k − x^k||_2 ≤ 10^−7 ou |∇f(x^k)^T(y^k − x^k)| ≤ 10^−7, então
4  |  Faça x* ← x^k e vá para a linha 10, pois estamos no ótimo;
5  senão
6  |  Acrescente a C a coluna correspondente a x^k;
7  |  Resolva (5.6), encontrando os pesos ótimos w*;
8  |  Calcule x^{k+1} de acordo com (5.7);
9  |  Faça k ← k + 1 e volte para a linha 2;
10 fim
```

Nele, inicialmente encontramos um ponto extremo de (5.3) (linha 1), para depois encontrar a solução do subproblema de PLI (linha 2) e testar se a solução atual satisfaz os critérios de parada (linha 3). Caso satisfaça, paramos o algoritmo e determinamos que a solução ótima é a combinação convexa dos pontos extremos já conhecidos que minimiza o valor da função objetivo do problema (linha 4). Senão, atualizamos a matriz C, cujas colunas guardam os pontos extremos conhecidos do envoltório convexo (linha 6), resolvemos o Problema (5.6), determinando o valor ótimo dos

pesos da combinação convexa que minimiza o valor de $f(x)$ (linha 7), atualizamos a solução atual do problema (linha 8) e voltamos para a linha 2 (linha 9).

5.2 Melhorando a solução obtida pelo CHR

Uma vez que o Algoritmo CHR encontra o ótimo de uma relaxação contínua do Problema (5.2), não temos nenhuma garantia que a solução obtida será inteira. Entretanto, o método, além de ser de fácil implementação, consegue um bom limitante inferior para o valor ótimo do problema, já que ele relaxa todas as restrições sobre seu envoltório convexo. Além disso, conforme se pode ver no trabalho de Guignard e Ahlatçioglu [26], o método mostrou-se barato computacionalmente e numericamente estável para problemas com um grande número de variáveis.

Como a solução de (3.1) obtida pelo CHR só é aplicável quando ela for inteira, foi criada uma heurística que tenta encontrar uma solução inteira próxima da solução fracionária obtida pelo método CHR.

A ideia da heurística consiste em, primeiramente, determinarmos o índice das K maiores variáveis z da solução do Problema (3.1) determinada pelo CHR. Caso tenhamos menos de K variáveis z não nulas, determinamos o índice de todas elas. Chamaremos de id o índice dessas variáveis. Logo após o primeiro passo, atribuímos o valor 1 para as variáveis z associadas a id e o valor 0 para as variáveis z cujo índice não estão em id. O problema resultante é um problema quadrático equivalente ao Subproblema (3.2), em que as variáveis x do problema possuem os índices do conjunto id. Esse problema pode ser resolvido facilmente por qualquer método clássico de programação quadrática. Após esse processo, verifica-se se o valor da função objetivo calculado na solução do problema quadrático, que nesse capítulo será denotado por ζ, é o mesmo valor ótimo que o método CHR obteve. Nesse caso, a solução, que é inteira por construção, também é ótima para o Problema (3.1). Caso contrário, nada se pode afirmar sobre a solução obtida. Esse procedimento pode ser justificado pelo fato do CHR fornecer um limitante inferior para o valor ótimo do problema. Logo, caso uma solução inteira atinja-o, ela será ótima para (3.1).

5.2.1 A Heurística CHR

Por meio das ideias apresentadas nessa seção, implementamos computacionalmente a heurística que tenta "corrigir" a solução do CHR, que será chamada de *Heurística CHR*, e é a nossa segunda contribuição nesse trabalho. Seu pseudocódigo é apresentado no Algoritmo 4.

Nessa heurística, no passo inicial encontramos a solução do problema via o Algoritmo CHR (linha 1) e testamos se ela é inteira (linha 2). Caso não o seja, calculamos o conjunto id (linhas 3 a

Algoritmo 4: A Heurística CHR

1 Resolva o Problema (3.1) pelo Algoritmo CHR, obtendo como solução os vetores de variáveis x^1 e z^1 e o valor ótimo ζ^1;
2 **se** *a solução obtida pelo Algoritmo CHR não for inteira,* **então**
3 | **se** *o vetor* z^1 *possuir mais de* K *componentes não nulas,* **então**
4 | | Faça id receber o índice das K maiores componentes de z^1;
5 | **senão**
6 | | Faça id receber o índice das variáveis não nulas de z^1;
7 | **fim**
8 | Faça $z^2(i) = \begin{cases} 1, \text{se } i \in id \\ 0, \text{se } i \notin id. \end{cases}$;
9 | Aplique os valores de z^2 em (3.1) e crie um problema do tipo (3.2);
10 | Resolva o problema resultante, encontrando a solução x^2 e o valor $\zeta^2 = f(x^2)$;
11 | **se** $\zeta^1 = \zeta^2$, **então**
12 | | $x^* \leftarrow x^2$ e $z^* \leftarrow z^2$;
13 | **senão**
14 | | A heurística não encontrou a solução ótima do Problema (3.1)
15 | **fim**
16 **senão**
17 | $x^* \leftarrow x^1$ e $z^* \leftarrow z^1$;
18 **fim**

7), fixamos o vetor z de acordo com esse conjunto (linha 8), criamos um problema de programação quadrática com base nele (linha 9) e o resolvemos (linha 10). Se o valor da função objetivo calculado na solução desse problema for o mesmo daquele calculado na linha 1, determinamos que a solução do problema de programação quadrática é ótima para (3.1) (linha 12), senão o algoritmo termina sem determinar o ótimo do problema (linha 14).

5.2.2 Um exemplo de aplicação da Heurística CHR

Consideremos o Problema (3.1) com $n = 3$, $K = 2$, $\lambda = 3{,}007$. Tomemos como dados iniciais:

$$Q = \begin{bmatrix} 0{,}1876 & 0{,}2910 & 0{,}0685 \\ 0{,}2910 & 1{,}2346 & 0{,}5953 \\ 0{,}0685 & 0{,}5953 & 0{,}4270 \end{bmatrix}, \mu = \begin{bmatrix} 0{,}1343 \\ 0{,}0986 \\ 0{,}1420 \end{bmatrix}, l = \begin{bmatrix} 0{,}2 \\ 0{,}2 \\ 0{,}2 \end{bmatrix}.$$

Mostramos abaixo, passo a passo, a aplicação da heurística.

- Linha 1: A solução obtida pelo CHR é $x^1 = \begin{bmatrix} 0{,}7021 \\ 0 \\ 0{,}2979 \end{bmatrix}$ e $z^1 = \begin{bmatrix} 0{,}7021 \\ 0 \\ 0{,}2979 \end{bmatrix}$, com $\zeta^1 = -0{,}3312$.
- Linha 2: A solução obtida pelo CHR não é inteira, então vamos para a linha 3.
- Linha 3: z^1 possui exatamente $K = 2$ componentes não nulas, então vamos para a linha 6.
- Linha 6: $id = \{1{,}3\}$.
- Linha 8: As componentes não nulas de z^2 são a primeira e a terceira, logo $z^2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.
- Linha 9: O novo problema, na forma (3.2), possui os seguintes parâmetros alterados:

$$Q = \begin{bmatrix} 0{,}1876 & 0{,}0685 \\ 0{,}0685 & 0{,}4270 \end{bmatrix}, \mu = \begin{bmatrix} 0{,}1343 \\ 0{,}1420 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} b = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$$

- Linha 10: A solução do problema nos fornece $x^2 = \begin{bmatrix} 0{,}7021 \\ 0 \\ 0{,}2979 \end{bmatrix}$ e $\zeta^2 = -0{,}3312$.
- Linha 11: Como $\zeta^1 = \zeta^2$, vamos para a linha 12.
- Linha 12: $x^* = \begin{bmatrix} 0{,}7021 \\ 0 \\ 0{,}2979 \end{bmatrix}$ e $z^* = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.
- Linha 18: Fim.

Nesse caso, a parte em x da solução obtida pelo CHR já era ótima, porém o vetor z tornava a solução infactível. A Heurística CHR manteve a solução ótima em x e integralizou as variáveis z, conseguindo encontrar a solução inteira ótima.

5.3 Um algoritmo exato para encontrar a solução do problema com variáveis binárias

Uma vez que o CHR nem sempre obtém a solução ótima de (3.1), foi necessário dispor de um método direto que convirja sempre à solução exata do problema. Como o Modelo (3.1) envolve variáveis binárias, uma maneira de se encontrar a solução do mesmo, para um valor fixo de λ, consiste em aplicar o método *Branch-and-Bound* para variáveis 0-1. Mais informações sobre esse algoritmo podem ser encontradas no Apêndice (A.1).

Como problemas desse tipo são em geral NP-difíceis, o tempo computacional para resolvê-los pode se tornar elevado. De forma a otimizar o nosso algoritmo principal, procuramos métodos similares, que poderiam acelerar o processo. Os métodos investigados foram:

- O uso do algoritmo *Branch-and-Cut*, que inclui no problema restrições baseadas em métodos de planos de corte, sempre que possível. Seguindo essa linha, Bienstock [6] investigou quatro estratégias diferentes de geração de planos de corte para o problema: cortes arredondados inteiro-mistos, cortes de mochila, cortes de interseção e cortes disjuntos. Dentre elas, a última se mostrou mais eficiente, devido ao fato da separação ser feita de maneira direta.

- O uso do algoritmo *Branch-and-Bound* implícito, combinado com o método de Lemke, para a resolução dos subproblemas quadráticos, além de técnicas de reformulação e eliminação de variáveis. Essa estratégia, de autoria de Bertsimas e Shioda [4], foi usada por Villela [36] em 2008.

Dentre as estratégias descritas, optamos pela segunda, uma vez que o método de Lemke pode ser facilmente adaptado à resolução dos problemas gerados a cada iteração do *Branch-and-Bound*. Problemas estes que possuem, geralmente, um número menor de restrições e para os quais dispomos de uma solução inicial muito boa. Nas subseções a seguir, vamos explicar o funcionamento dessa estratégia.

5.3.1 Trabalhando com o problema relaxado

A cada passo do algoritmo *Branch-and-Bound* implícito, precisamos resolver um problema relaxado, do qual excluímos as variáveis inteiras e as restrições que as envolvem. O problema

obtido possui a seguinte forma:

$$\begin{array}{rl} \min & \frac{1}{2}x^T Qx + c^T x \\ S.a & Ax \leq b \\ & x \geq 0. \end{array} \tag{5.8}$$

Como Q é uma matriz simétrica definida positiva, as condições de Karush-Kuhn-Tucker são necessárias e suficientes para a caracterização de um mínimo global do problema. Tais condições são definidas por

$$\begin{cases} c + Qx + g^T A - d = 0 \\ g^T(Ax - b) = 0 \\ -d^T x = 0 \\ x,g,d \geq 0, \end{cases}$$

onde, $g \in \mathbb{R}^n$ e $d \in \mathbb{R}^m$ são denominadas *variáveis duais*. Com o uso de um vetor auxiliar $v \in \mathbb{R}^m$, podemos reescrever as condições acima como

$$\begin{cases} d = c + Qx + A^T g \\ v = b - Ax \\ d^T x = 0, v^T g = 0 \\ x,g,d,v \geq 0. \end{cases} \tag{5.9}$$

Utilizando a representação

$$q = \begin{bmatrix} c \\ b \end{bmatrix}, M = \begin{bmatrix} Q & A^T \\ -A & 0 \end{bmatrix}, \psi = \begin{bmatrix} x \\ g \end{bmatrix}, \omega = \begin{bmatrix} d \\ v \end{bmatrix},$$

é possível reescrever (5.9) na forma

$$\begin{cases} \omega = M\psi + q \\ \omega^T \psi = 0 \\ \omega,\psi \geq 0. \end{cases} \tag{5.10}$$

O Problema (5.8), escrito na forma (5.10), é conhecido como problema de complementaridade linear, que chamaremos de *PCL(q,M)*, e pode ser resolvido por vários métodos clássicos dedicados a esse assunto. Mais detalhes sobre esse tipo de problema podem ser encontrados no Apêndice A.2.

Dentre as diversas alternativas existentes para a resolução de (5.10), optamos por utilizar o

método de Lemke, pois ele é particularmente eficiente quando começamos de uma solução infactível próxima à solução ótima, situação comumente encontrada durante a aplicação do método *Branch-and-Bound* implícito, como será visto na Subseção 5.3.3. Na subseção a seguir, vamos falar mais sobre esse método.

5.3.2 O método de Lemke

Dados q, M e um vetor h positivo definido pelo usuário, o método de Lemke resolve o *PCL(q,M)* por meio de uma série de pivoteamentos. Primeiramente, verificamos se $q \geq 0$, pois, nesse caso, temos a solução trivial $\psi = 0$ e $\omega = q$ como solução do problema. Caso isso não aconteça, precisamos definir uma base complementar inicial. As variáveis básicas, entretanto, não precisam satisfazer a restrição de não negatividade. A estratégia padrão, nesse caso, é fazer com que as componentes de ψ sejam não básicas e as de ω sejam básicas, de modo que a base inicial seja a matriz identidade.

Em seguida, aumentamos o *PCL(q,M)*, substituindo (5.10) por

$$\begin{cases} \omega = M\psi + q + h\psi_o \\ \psi \geq 0, \omega \geq 0, \psi_0 \geq 0 \\ \psi^T \omega = 0. \end{cases}$$

A ideia implícita no aumento do PCL é a de que, ao introduzirmos no problema a coluna positiva h relativa a ψ_0 e forçarmos tal variável a entrar na base, tornamos a nossa solução básica positiva. Para mostrar que isso acontece, consideremos que a base inicial seja a trivial. Neste caso, queremos que $\omega \geq 0$, o que implica em $q + h\psi_0 \geq 0$, pois $\psi = 0$. Escolhendo o vetor h de forma adequada e definindo

$$\psi_0 = max \left\{ \frac{-q_i}{h_i} \right\},$$

garantimos que a solução obtida após a transformação de ψ_0 em variável básica seja necessariamente positiva. Nesse caso, a variável que é forçada a sair da base para a inclusão de ψ_0 é exatamente aquela definida por $argmax \{-q_i/h_i\}$.

A introdução de ψ_0 faz com que um par (ψ_i, ω_i) deixe de ser complementar, pois as duas variáveis que o compõem passam a ser não básicas. Para corrigir esse problema, o método de Lemke escolhe, a cada iteração, a variável complementar àquela que saiu da base na iteração anterior para entrar na base da iteração atual.

A variável que entra na base pode ter seu valor aumentado, até que uma das variáveis básicas

se torne zero. A primeira variável básica a atingir o valor zero (denominada variável de bloqueio) deixa a base. O processo de determinação da variável que sai da base é análogo ao teste da razão, utilizado na programação linear.

Repetimos o processo de substituição de variáveis na base até que ψ_0 se torne a variável de bloqueio, ou até que não consigamos achar uma variável de bloqueio. No primeiro caso, ψ_0 é retirada da base e, assim, obtemos uma solução que é ao mesmo tempo factível e complementar, de modo que dispomos da solução ótima. No segundo caso, o nosso PCL original (5.10) (e, consequentemente, nosso problema de otimização quadrática) é infactível.

5.3.3 O método *Branch-and-Bound* implícito

O algoritmo *Branch-and-Bound* implícito resolve problemas quadráticos do tipo

$$\begin{array}{rl} \min & \frac{1}{2}x^TQx + c^Tx \\ S.a & Ax \leq b \\ & Card(nz(x)) \leq K \\ & x_i \geq l_i, i \in nz(x) \\ & x_i = 0, i \notin nz(x), \end{array} \tag{5.11}$$

que pode ser entendido como a versão geral do Problema (2.6), apresentado na Seção 2.4, ou uma formulação alternativa do Problema (5.1).

A principal diferença entre essa formulação e a de um problema de programação quadrática 0-1 convencional é que as variáveis binárias estão implícitas, ou seja, não aparecem, mas são necessárias para que tenhamos no máximo K variáveis positivas, ou seja, para que

$$Card(nz(x)) \leq K. \tag{5.12}$$

Essa forma implícita de tratar as variáveis binárias pode ser estendida ao método *Branch-and-Bound* para variáveis 0-1. Neste caso, eliminamos do problema relaxado (o nó raiz da árvore binária) a Restrição (5.12) e os limites

$$\begin{array}{l} x_i \geq l_i, i \in nz(x) \\ x_i = 0, i \notin nz(x), \end{array} \tag{5.13}$$

adotando apenas a restrição de não negatividade $x \geq 0$.

Obtemos, assim, um problema clássico de programação quadrática, na forma (5.8). A única diferença entre este problema e o correspondente ao nó raiz do *Branch-and-Bound* usual é a

inexistência da restrição $\sum_{i=1}^{n} z_i \leq K$ presente na formulação (5.1), mas não em (5.11).

A relaxação das restrições de integralidade das variáveis binárias, permite que a cardinalidade de *nz(x)* seja superior a K e que as componentes estritamente positivas do vetor x assumam valores abaixo de seus limites inferiores. No caso provável da solução do nó raiz da árvore não satisfazer as restrições (5.12) ou (5.13), é necessário ramificar uma variável qualquer. É justamente na manipulação das ramificações que as restrições ignoradas são levadas em conta, justificando o caráter implícito do método *Branch-and-Bound*.

A ideia é fazer com que, a cada ramificação, as variáveis estejam mais próximas de obedecer às restrições de cardinalidade e limite inferior. Usualmente, a ramificação do método *Branch-and-Bound* é feita sobre as variáveis binárias, z_i. Porém, nessa formulação, essas variáveis estão definidas de forma implícita, de forma que a ramificação também será feita desse jeito, considerando os efeitos que a ramificação de uma variável binária z_i, provoca sobre a sua variável real associada, x_i. Vejamos como ramificar, de maneira implícita, uma variável z_s:

- Habitualmente, a ramificação para baixo corresponde à atribuição do valor 0 à variável binária associada a x_s, obrigando-nos a adotar $x_s = 0$. No método implícito, essa ramificação é obtida por meio da eliminação direta de x_s, o que provoca a geração de um novo subproblema com uma variável a menos. Esse subproblema é resolvido pelo do método de Lemke.

- No problema financeiro usual, a ramificação para cima está relacionada à inclusão forçada de um ativo na carteira. No *Branch-and-Bound* para problemas 0-1, isso equivale a atribuir o valor 1 à variável binária associada à variável x_s, ao passo que, no *Branch-and-Bound* implícito, tal exigência se resume à inclusão da restrição $x_s \geq l_s$. Esse tipo de ramificação traz consequências não só à restrição de limite inferior, mas também à restrição de cardinalidade, que pode passar a ser infactível. No método implícito, essa ramificação é feita apenas quando x_s é estritamente positiva e está abaixo do seu limite inferior l_s. Neste caso, incluímos no problema a restrição $x_s \geq l_s$ e o resolvemos pelo método de Lemke. Para evitar que a Restrição (5.12) se torne infactível, a ramificação para cima não é feita se o número de variáveis ramificadas para cima for superior ou igual a K.

É possível observar que o problema gerado pelas ramificações é muito parecido com o original, em geral diferindo apenas por uma restrição ou pelo número de variáveis. Dessa forma, a solução do problema que gera as ramificações é ótima para os problemas ramificados, porém infactível. Como o método de Lemke resolve problemas de complementaridade linear, fica evidente que, partindo de soluções infactíveis, ele pode resolver o problema gerado pelas ramificações em poucos passos, já que a solução inicial é quase ótima, gerando um processo que é conhecido como *warm-start*.

Para facilitar a compreensão da versão implícita do *Branch-and-Bound*, apresentamos os passos que compõem o algoritmo (que chamaremos de *Algoritmo BBI*) por meio de um pseudocódigo que está ilustrado no Algoritmo 5.

Algoritmo 5: O Algoritmo BBI

```
1  Utilizando o método de Lemke, resolva o problema relaxado, retirando as Restrições (5.12) e
   (5.13);
2  se na solução obtida, não tivermos mais que K variáveis não nulas e se estas satisfizerem
   as restrições de limite inferior, então
3  |  Pare, pois a solução do problema relaxado já é ótima;
4  senão
5  |  Escolha uma variável binária não inteira para ramificar;
6  |  Crie uma lista de nós pendentes, nosp, contendo os dois subproblemas gerados pela
   |  ramificação;
7  |  x* ← [ ] (vetor indefinido), ζ* ← +∞;
8  |  enquanto nosp ≠ ∅ faça
9  |  |  Retire um subproblema de nosp;
10 |  |  Usando o método de Lemke, resolva esse subproblema, obtendo x e ζ ou a indicação
   |  |  de que ele é infactível;
11 |  |  se x possuir componentes não nulas menores que os limites inferiores
   |  |  correspondentes ou se o número de componentes não nulas for maior que K, então
12 |  |  |  Escolha uma variável para ramificar;
13 |  |  |  Inclua em nosp o problema relacionado à ramificação para baixo;
14 |  |  |  se a ramificação para cima não violar a Restrição (5.12), então
15 |  |  |  |  Inclua a ramificação para cima em nosp;
16 |  |  |  fim
17 |  |  senão
18 |  |  |  se ζ < ζ*, então
19 |  |  |  |  x* ← x e ζ* ← ζ;
20 |  |  |  |  para todo nó i de nosp faça
21 |  |  |  |  |  se ζ_i^p ≥ ζ*, onde ζ_i^p é o valor da função objetivo do pai do nó i, então
22 |  |  |  |  |  |  Elimine o nó i de nosp;
23 |  |  |  |  |  fim
24 |  |  |  |  fim
25 |  |  |  fim
26 |  |  fim
27 |  fim
28 |  A solução do problema é x*, com função objetivo igual a ζ*;
29 fim
```

Nesse algoritmo, primeiramente resolvemos o problema relaxado via o método de Lemke (linha

1) e determinamos se sua solução é ótima (linha 2). Caso não o seja, o que é natural de se esperar, escolhemos uma variável para ramificar e criamos uma lista inicial de nós pendentes (linhas 5 e 6). Na linha 8, começamos a parte iterativa do método, que acaba quando todos os subproblemas relevantes forem resolvidos.

Em uma iteração qualquer, retiramos um subproblema da lista e o resolvemos pelo método de Lemke (linhas 9 e 10). Caso a solução obtida não for factível para o Problema (3.1), geramos, se possível, dois nós filhos, provenientes desse subproblema, e os incluímos na lista de nós pendentes (linhas 12 a 15). Senão, ela é inteira e factível para (3.1), de modo que podemos testar se a mesma é melhor que a atual. Nesse caso, atualizamos a solução do problema (linha 19) e eliminamos, de nossa lista, os nós filhos cujos pais são piores que ela (linhas 21 a 23).

5.3.4 Um Exemplo

Consideremos o Problema (5.11) com $K = 2$. Tomemos como dados iniciais:

$$Q = \begin{bmatrix} 1{,}1 & 1 & 1 \\ 1 & 1{,}25 & 1 \\ 1 & 1 & 1{,}3 \end{bmatrix}, c = \begin{bmatrix} -0{,}935 \\ -1{,}075 \\ -1{,}3 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, b = 1, l = \begin{bmatrix} 0{,}3 \\ 0{,}5 \\ 0{,}85 \end{bmatrix}.$$

Ao longo do algoritmo, representaremos cada subproblema da lista de nós pendentes (nosp) utilizando o vetor *ram*, cuja *i-ésima* coordenada é definida conforme indicado abaixo:

- $ram(i) = -1$, se a variável x_i ainda não tiver sido ramificada;
- $ram(i) = 0$, se a variável x_i tiver sido ramificada para baixo;
- $ram(i) = 1$, se a variável x_i tiver sido ramificada para cima.

Mostramos abaixo, passo a passo, a aplicação do algoritmo.

- Linha 1: Resolvendo o nó raiz, obtemos $\zeta = -1{,}1167$ e $x = [\ 0 \quad 0{,}364 \quad 0{,}62\ ]^T$.
- Linha 2: Como as variáveis 2 e 3 estão abaixo do limite inferior e acima de zero, então vamos para a linha 5.
- Linha 5: Escolhemos x_3 para ramificar, apesar da escolha da segunda variável também ser plausível.
- Linha 6: Entram na lista de nós pendentes as ramificações para cima e para baixo de x_3. Assim, $nosp = \left\{[\ -1 \quad -1 \quad 0\ ]^T, [\ -1 \quad -1 \quad 1\ ]^T\right\}$.

- Linha 8: Aqui começamos a parte iterativa do método.
  - Iteração 1:
  - Linha 9: Retiramos o subproblema $[\ -1 \quad -1 \quad 0\ ]^T$ de *nosp*. Logo, $nosp = \left\{[\ -1 \quad -1 \quad 1\ ]^T\right\}$.
  - Linha 10: Resolvendo o subproblema, obtemos $\zeta = -0{,}9432$ e $x = [\ 0{,}25 \quad 0{,}66 \quad 0\ ]^T$.
  - Linha 11: A componente 1 do vetor x está abaixo de seu limite inferior, então vamos para a linha 12.
    - Linha 12: Escolhemos x_1 para ramificar, pois esta é a única variável não nula abaixo de seu limite inferior.
    - Linha 13: Incluímos em *nosp* o nó correspondente à ramificação para baixo de x_1. Logo, $nosp = \left\{[\ -1 \quad -1 \quad 1\ ]^T, [\ 0 \quad -1 \quad 0\ ]^T\right\}$.
    - Linha 14: A ramificação para cima de x_1 não viola (5.12), então vamos para a linha 15.
    - Linha 15: Incluímos em *nosp* o nó correspondente à ramificação para cima de x_1. Assim, $nosp = \left\{[\ -1 \quad -1 \quad 1\ ]^T, [\ 0 \quad -1 \quad 0\ ]^T, [\ 1 \quad -1 \quad 0\ ]^T\right\}$.
  - Iteração 2:
  - Linha 9: Retiramos o subproblema $[\ -1 \quad -1 \quad 1\ ]^T$ de *nosp*. Logo, $nosp = \left\{[\ 0 \quad -1 \quad 0\ ]^T, [\ 1 \quad -1 \quad 0\ ]^T\right\}$.
  - Linha 10: Resolvendo o subproblema, obtemos $\zeta = -1{,}0891$ e $x = [\ 0 \quad 0{,}015 \quad 0{,}85\ ]^T$.
  - Linha 11: A componente 2 do vetor x está abaixo de seu limite inferior, então vamos para a linha 12.
    - Linha 12: Escolhemos x_2 para ramificar, pois esta é a única variável não nula abaixo de seu limite inferior.
    - Linha 13: Incluímos em *nosp* o nó correspondente à ramificação para baixo de x_2. Logo, $nosp = \left\{[\ 0 \quad -1 \quad 0\ ]^T, [\ 1 \quad -1 \quad 0\ ]^T, [\ -1 \quad 0 \quad 1\ ]^T\right\}$.
    - Linha 14: A ramificação para cima de x_2 viola (5.12). Logo, não a incluímos em *nosp*.
  - Iteração 3:
  - Linha 9: Retiramos o subproblema $[\ -1 \quad 0 \quad 1\ ]^T$ de *nosp*. Logo, $nosp = \left\{[\ 0 \quad -1 \quad 0\ ]^T, [\ 1 \quad -1 \quad 0\ ]^T\right\}$.

- Linha 10: Resolvendo o subproblema, obtemos $\zeta = -1{,}0563$ e $x = [\ 0{,}0773 \quad 0 \quad 0{,}85\ ]^T$.
- Linha 11: A componente 1 do vetor x está abaixo de seu limite inferior, então vamos para a linha 12.
  - ∗ Linha 12: Escolhemos x_1 para ramificar, pois esta é a única variável não nula abaixo de seu limite inferior.
  - ∗ Linha 13: Incluímos em *nosp* o nó correspondente à ramificação para baixo de x_1. Logo, $nosp = \left\{[\ 0 \quad -1 \quad 0\ ]^T, [\ 1 \quad -1 \quad 0\ ]^T, [\ -1 \quad 0 \quad 1\ ]^T, [\ 0 \quad 0 \quad 0\ ]^T\right\}$.
  - ∗ Linha 14: A ramificação para cima de x_1 viola (5.12). Logo, não a incluímos em *nosp*.
- Iteração 4:
- Linha 9: Retiramos o subproblema $[\ 0 \quad 0 \quad 1\ ]^T$ de *nosp*. Logo, $nosp = \left\{[\ 0 \quad -1 \quad 0\ ]^T, [\ 1 \quad -1 \quad 0\ ]^T\right\}$.
- Linha 10: Resolvendo o subproblema, obtemos $\zeta = -1{,}053$ e $x = [\ 0 \quad 0 \quad 0{,}9\ ]^T$.
- Linha 11: Como x não possui componente não nula menor que os seus limites inferiores e não viola a restrição de cardinalidade, encontramos uma solução inteira mista factível. então então vamos para a linha 18.
- Linha 18: Observamos que $\zeta < \zeta^*$, pois esta é a primeira solução factível encontrada, então vamos para a linha 19.
  - ∗ Linha 19: $x^* \leftarrow [\ 0 \quad 0 \quad 0{,}9\ ]^T$ e $\zeta^* \leftarrow -1{,}053$
  - ∗ Linhas 20-24: Como $\zeta^p \geq \zeta^*$ para os nós da lista de nós pendentes, estes são eliminados e $nosp = \emptyset$.
- Linha 28: A solução é $x^* = [\ 0 \quad 0 \quad 0{,}9\ ]^T$ e $\zeta^* = -1{,}053$.

- Linha 29: Fim.

A Figura 5.3 mostra a árvore explorada durante o exemplo.

5.4 Combinando os métodos CHR e BBI

Uma vez que o método CHR resolve uma relaxação de um problema com variáveis inteiras, a solução obtida pelo mesmo pode não ser inteira, mesmo após a aplicação da Heurística CHR. Testes mostraram que a solução frequentemente deixa de ser inteira para valores de λ próximos

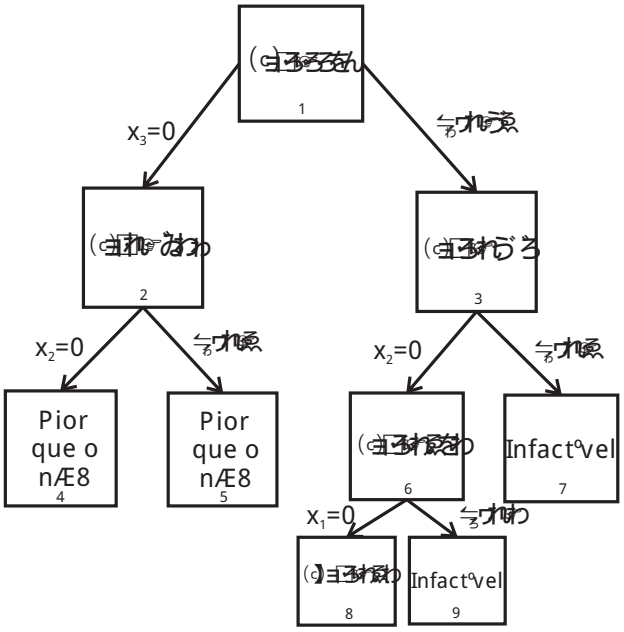


Figura 5.3: Árvore binária gerada pelo algoritmo *Branch-and-Bound* implícito.

de 0, ou seja, quando a aversão ao risco é alta. Apesar desse inconveniente, o método mostrou-se muito barato durante os testes realizados em problemas de grande porte, principalmente para valores altos de λ, quando ele superou drasticamente, em tempo computacional, o Algoritmo BBI. A vantagem deste último é que, apesar de ser usualmente mais caro, o mesmo sempre acha a solução ótima inteira do Problema (3.1). Como o Algoritmo BBI é capaz de aproveitar o valor da função objetivo do nosso problema, calculado na solução obtida pela Heurística CHR, utilizando-o como limitante superior, parece ser interessante combinar esses dois métodos, pois o segundo pode ser acelerado pela solução quase ótima do primeiro.

Seguindo esse raciocínio, criamos o Algoritmo PQIM, que utiliza essas estratégia para combinar os dois métodos de forma eficiente, de modo a encontrar a solução de (3.1) com λ fixo. Seu pseudocódigo é apresentado no Algoritmo 6

Tal algoritmo, resolve, primeiramente, o problema pelo CHR (linha 1) e, quando a solução obtida não é inteira, tenta corrigi-la por meio da Heurística CHR (linha 3). Caso a solução obtida ainda não seja a procurada, aplica-se o Algoritmo BBI, utilizando o valor da função objetivo do nosso problema, calculado na solução obtida pela Heurística CHR, como limitante superior inicial (linha 5). Esse aproveitamento pode ser entendido como a terceira de nossas contribuições.

Algoritmo 6: O Algoritmo PQIM

```
1  Resolva o Problema (3.1) pelo do Algoritmo CHR;
2  se a solução obtida pelo CHR não for inteira, então
3  |  Corrija a solução pela Heurística CHR;
4  |  se a solução obtida no passo anterior não for ótima, então
5  |  |  Resolva o problema pelo Algoritmo BBI, tomando o valor da função objetivo,
   |  |  calculado na solução obtida pela Heurística CHR, como limitante superior;
6  |  senão
7  |  |  A solução obtida pela Heurística CHR é ótima;
8  |  fim
9  senão
10 |  A solução obtida pelo Algoritmo CHR é ótima;
11 fim
```

5.4.1 Um exemplo numérico do Algoritmo PQIM

Considere o Problema (3.1), com $\lambda = 40{,}3558$ e os mesmos parâmetros apresentados na Subseção 5.2.2.

Mostramos abaixo, passo a passo, a aplicação do algoritmo.

- Linha 1: A solução obtida pelo CHR é $x^1 = \begin{bmatrix} 0{,}1 \\ 0 \\ 0{,}9 \end{bmatrix}$ e $z^1 = \begin{bmatrix} 0{,}1 \\ 0 \\ 0{,}9 \end{bmatrix}$ com $\zeta_1 = -5{,}5194$.
- Linha 2: A solução obtida não é inteira, pois z possui componentes fracionárias, então vamos para a linha 3.
- Linha 3: A solução obtida pela Heurística CHR é $x^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ e $z^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ com $\zeta_2 = -5{,}5170$.
- Linha 4: Como $\zeta_1 \neq \zeta_2$, a solução obtida pela Heurística CHR não resolve o nosso problema, então vamos para a linha 5.
- Linha 5: A solução obtida pelo Algoritmo BBI é $x^* = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ e $z^* = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.
- Linha 11: Fim

Note que, nesse caso, a solução obtida pela Heurística CHR já era ótima. Entretanto, não foi possível perceber esse fato sem aplicarmos o Algoritmo BBI. Apesar disso, o Algoritmo BBI verificou rapidamente que a solução obtida pela Heurística CHR era ótima, pois ele utiliza o valor ótimo dela como limitante superior.

Capítulo 6

O Algoritmo PFV

Este é o capítulo principal da tese, no qual vamos apresentar o Algoritmo das Interseções, que determina o encontro de duas curvas associadas a carteiras distintas, assim como a proposta para aproximar a FEI do Problema (3.1) (Algoritmo PFV), por meio da *combinação* desse algoritmo com aquele que caminha sobre um seguimento de parábola, em ambos os sentidos de trajetória (Algoritmo Segmento FEI), e o que resolve um problema de programação quadrática inteira mista (Algoritmo PQIM).

6.1 Parametrizando um segmento em que o conjunto ativo é constante

Na Seção (4.3), descrevemos um algoritmo que determina um segmento da FEI do Problema (3.2), relativo a um conjunto ativo constante. Nesse caso, o segmento é caracterizado pelo intervalo de λ em que o conjunto ativo está fixado e pela solução de (3.2) nos extremos. De posse dessas informações, é possível definir, nesse intervalo, em função de λ, a solução geral de (3.2), assim como o valor da função objetivo desse problema.

Seja s um segmento qualquer da FEI em que um conjunto ativo I_a está fixado e $[\lambda_1,\lambda_2]$ o intervalo de λ correspondente. Suponha, ainda, que conheçamos as soluções de (3.2), x^1 e x^2, associadas às extremidades inicial e final desse segmento. Qualquer solução de (3.2), no segmento s, pode ser ser escrita em função de $\lambda \in [\lambda_1,\lambda_2]$, pela seguinte combinação linear:

$$x(\lambda) = \frac{\lambda - \lambda_1}{\lambda_2 - \lambda_1} x^2 + \frac{\lambda_2 - \lambda}{\lambda_2 - \lambda_1} x^1.$$

Observe que $x(\lambda_1) = x^1$ e $x(\lambda_2) = x^2$. Desenvolvendo essa expressão, obtemos:

$$x(\lambda) = \frac{1}{\lambda_2 - \lambda_1}(\lambda x^2 - \lambda_1 x^2 + \lambda_2 x^1 - \lambda x^1) = \frac{\lambda_2 x^1 - \lambda_1 x^2}{\lambda_2 - \lambda_1} + \frac{x^2 - x^1}{\lambda_2 - \lambda_1}\lambda = v_1 + \lambda v_2, \tag{6.1}$$

onde $v_1 = \frac{\lambda_2 x^1 - \lambda_1 x^2}{\lambda_2 - \lambda_1}$ e $v_2 = \frac{x^2 - x^1}{\lambda_2 - \lambda_1}$.

Reescrevendo a função objetivo de (3.2) em função de λ, obtemos:

$$\begin{aligned} f(\lambda) &= \tfrac{1}{2}x(\lambda)^t Q x(\lambda) - \lambda\mu^t x(\lambda) = \tfrac{1}{2}(v_1 + \lambda v_2)^t Q(v_1 + \lambda v_2) - \lambda\mu^t(v_1 + \lambda v_2) \\ &= \tfrac{1}{2}\left(v_1^t Q v_1 + \lambda v_1^t Q v_2 + \lambda v_2^t Q v_1 + \lambda^2 v_2^t Q v_2\right) - \lambda\mu^t v_1 - \lambda^2\mu^t v_2 \\ &= \left(\tfrac{1}{2}v_2^t Q v_2 - \mu^t v_2\right)\lambda^2 + \left(v_1^t Q v_2 - \mu^t v_1\right)\lambda + \tfrac{1}{2}v_1^t Q v_1 = a\lambda^2 + b\lambda + c, \end{aligned} \tag{6.2}$$

onde

$$a = \frac{1}{2}v_2^t Q v_2 - \mu^t v_2,\ \ b = v_1^t Q v_2 - \mu^t v_1,\ \ c = \frac{1}{2}v_1^t Q v_1. \tag{6.3}$$

Dessa forma, a função objetivo de (3.2), no segmento s, pode ser parametrizada como uma função de segundo grau que depende de λ, ou seja, o trecho da FEI correspondente a s é um segmento de parábola. Naturalmente, essa parametrização pode ser estendida para todos os segmentos que compõem a FEI do Problema (3.2).

Como sabemos de antemão que a FEI é uma curva monótona e que a função objetivo de (3.2) decresce conforme λ aumenta, podemos afirmar que a curva $\lambda \times f(\lambda)$ é formada pela união não crescente de segmentos de parábola, conforme foi dito na Seção 3.3.

6.2 Encontrando a interseção de duas curvas compostas por carteiras distintas

A parametrização da curva $\lambda \times f(\lambda)$, referente ao Subproblema (3.2), pode ser estendida para cada uma das T curvas associadas à FEI do Problema (3.1), pois, conforme discutimos no Capítulo 3, cada uma delas possui o formato da curva de (3.2). Seguindo esse raciocínio, todas as curvas $\lambda \times f(\lambda)$ referentes ao Problema (3.1) podem ser parametrizadas e desenhadas em um mesmo sistema de eixos, conforme afirmamos na Seção 3.4.

Uma vez que podemos determinar, em função de $\lambda \in [0, +\infty)$, todas as curvas associadas a carteiras distintas, fica evidente que é possível encontrar analiticamente a interseção delas. Como a FEI do Problema (3.1) é a curva que está por baixo das demais, pode-se afirmar que ela contém segmentos de algumas das T curvas associadas a esse problema, ou seja, a curva que está por baixo

possui segmentos de curvas compostas por carteira distintas, e que a mudança de uma curva para outra ocorre na interseção das mesmas.

Como o Algoritmo PFV caminha sempre pela curva que está por baixo, precisamos de um algoritmo que encontre a interseção de duas curvas.

A Figura 6.1 ilustra uma situação típica de interseção entre duas curvas. Nela, a curva C_1, que possui um segmento de parábola s_1, definido em $[\lambda_{1,2},\lambda_{1,1}]$, intersecta duas vezes a curva C_2, que possui um segmento de parábola s_2, definido em $[\lambda_{2,2},\lambda_{2,1}]$. Os vetores $x^{1,1}$ e $x^{1,2}$ representam a solução do Problema (3.2), associado a s_1, nos extremos do intervalo em que esse segmento está definido. Da mesma forma, os vetores $x^{2,1}$ e $x^{2,2}$ representam a solução do Problema (3.2), associado a s_2, nos extremos em que ele está definido.

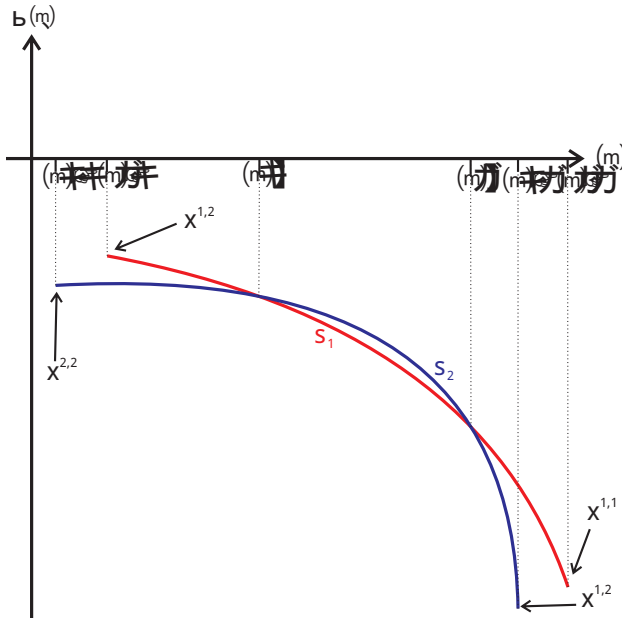


Figura 6.1: Interseção de duas curvas.

Na prática, para encontrar a interseção dessas duas curvas, estendemos o intervalo em que os segmentos s_1 e s_2 estão definidos para $\mathbb{R}$ e tentamos encontrar a coordenada λ em que esses segmentos se encontram. Caso essa coordenada esteja na interseção dos intervalos geradores de s_1 e s_2, podemos afirmar que ela representa a interseção de C_1 e C_2 nesse intervalo. Matematicamente, considere que $I_\lambda = [\lambda_{1,1},\lambda_{1,2}] \cap [\lambda_{2,1},\lambda_{2,2}]$ e que $f_1(\lambda)$ e $f_2(\lambda)$ sejam as funções parametrizadas dos segmentos s_1 e s_2, calculadas de acordo com (6.2), estendidas para todos os valores reais de λ. As possíveis coordenadas λ_1^* e λ_2^* em que $f_1(\lambda) = f_2(\lambda)$, representam a interseção dos dois segmentos estendidos. Caso $\lambda_1^* \in I_\lambda$ ou $\lambda_2^* \in I_\lambda$, pode-se afirmar que esses valores representam os cruzamentos de s_1 com s_2. Apesar de ser teoricamente possível, nunca ocorreram duas interseções em nossos

testes.

Por meio das ideias apresentadas nesse seção, criamos o Algoritmo das Interseções, a nossa quarta contribuição nesse trabalho, que calcula a interseção de duas curvas associadas à carteiras distintas, C_1 e C_2. Seu pseudocódigo é ilustrado no Algoritmo 7.

Algoritmo 7: O Algoritmo das Interseções

```
1  A partir dos parâmetros λ_{1,1}, λ_{1,2}, x^{1,1} e x^{1,2}, referentes ao segmento s_1, calcule, por meio de (6.3), os parâmetros a_1, b_1 e c_1;
2  A partir dos parâmetros λ_{2,1}, λ_{2,2}, x^{2,1} e x^{2,2}, referentes ao segmento s_2, calcule, por meio de (6.3), os parâmetros a_2, b_2, c_2;
3  Determine I_λ;
4  Faça a ← a_1 − a_2, b ← b_1 − b_2 e c ← c_1 − c_2;
5  Faça Δ ← b^2 − 4ac;
6  se Δ ≥ 10^{−8}, então
7  |  Faça λ̃_1 ← (−b+√Δ)/2a e λ̃_2 ← (−b−√Δ)/2a;
8  |  se λ̃_1 ∈ I_λ, então
9  |  |  Faça λ*_1 ← λ̃_1;
10 |  |  Faça f*_1 ← a(λ*_1)^2 + bλ*_1 + c;
11 |  fim
12 |  se λ̃_2 ∈ I_λ, então
13 |  |  Faça λ*_2 ← λ̃_2;
14 |  |  Faça f*_2 ← a(λ*_2)^2 + bλ* + c;
15 |  fim
16 |  se λ̃_1 ∉ I_λ e λ̃_2 ∉ I_λ, então
17 |  |  As curvas não se interceptam em I_λ;
18 |  fim
19 senão
20 |  As curvas não se interceptam em I_λ;
21 fim
```

O primeiro passo desse algoritmo consiste em calcular os parâmetro relativos às equações das parábolas que descrevem os segmentos s_1 e s_2 em I_λ (linhas 1 e 2) e o intervalo que vamos buscar a interseção desses segmentos (linha 3). Logo após, igualamos as equações das duas curvas, gerando uma equação de segundo grau, e determinamos os seus coeficientes (linha 4). Caso a equação tenha solução, ou seja, caso Δ seja não negativo, podemos garantir que os segmentos se interceptam, mesmo que fora do intervalo de busca (linha 6). Sendo assim, calculamos essas possíveis interseções (linha 7) e testamos se elas ocorrem dentro do intervalo I_λ, retornando aquelas que obedecerem esse teste (linha 8 a 15).

6.3 Encontrando uma solução inicial

A partir da combinação das três ferramentas descritas anteriormente (Algoritmo Segmento FEI , Algoritmo PQIM e Algoritmo das Interseções), é possível criar um método que aproxima a fronteira eficiente do Problema (3.1).

Para iniciá-lo, é necessário obter a solução inicial em uma das extremidades da fronteira eficiente, seja ela referente ao risco mínimo ($\lambda = 0$) ou ao retorno máximo ($\lambda \to +\infty$).

Uma vez que é permitida a existência de carteiras com um único ativo, a solução de (3.1) quando $\lambda \to +\infty$ consiste em investir 100% do capital na aplicação que possui o maior retorno, pois nessa situação o risco é completamente desconsiderado. Sendo assim, se $j = argmax\,\{\mu(i)\}$, temos para as variáveis x e z de (3.1):

$$x(i) = \begin{cases} 1, \text{se } i = j \\ 0, \text{se } i \neq j \end{cases} \quad e \; z(i) = \begin{cases} 1, \text{se } i = j \\ 0, \text{se } i \neq j. \end{cases}$$

Note que, a partir de um certo valor alto de λ, que chamaremos de λ_{max}, essa solução não se altera, ou seja, a função objetivo de (3.1) varia linearmente em função desse parâmetro, fazendo com que a extremidade direita da fronteira eficiente seja composta por uma reta decrescente.

Caso dois ou mais ativos possuam o maior retorno, investimos todo o nosso capital no de menor risco, seguindo um raciocínio análogo àquele apresentado na Seção 4.2. Uma vez que a solução de retorno máximo é inteira, ela é ótima para o Problema (3.1).

Seguindo um raciocínio semelhante, quando $\lambda \to 0$, apenas o risco é considerado. Como o risco é não negativo, devido ao fato de Q ser simétrica e definida positiva, e é permitida a ausência de investimento, o menor valor para o risco será zero, o que ocorre somente na solução nula ($x = z = \vec{0}$). É possível observar que, para valores de λ próximos de zero, a solução nula também é ótima, ou seja, a extremidade esquerda da fronteira eficiente é dada pela reta horizontal $f(\lambda) = 0$. A solução de risco mínimo é inteira, logo é ótima para o Problema (3.1).

Note que, a solução nula é singular, o que torna significativamente difícil a obtenção do segmento adjacente ao segmento nulo. Felizmente, essa singularidade é tratável de forma simples caso a situação seja inversa, i.e, quando conhecemos o segmento adjacente ao segmento nulo e queremos determinar o ponto de mudança de curva, situação que aparece somente quando caminhamos da direita para a esquerda na FEI do Problema (3.1). Explicaremos essa estratégia com mais detalhes na Subseção 6.8.1.

Para evitar a singularidade inicial, é mais vantajoso começar o método da extremidade direita da fronteira. Assim, a exemplo do problema com variáveis contínuas, começamos a resolver o problema com $\lambda \to +\infty$ e caminhamos da direita para a esquerda, até obtermos $\lambda = 0$.

Apesar de sabermos da existência de um valor λ_{max} a partir do qual a solução de (3.1) não muda, o Algoritmo PFV não precisa calculá-lo explicitamente, pois podemos calcular a solução de máximo retorno, que é aquela associada a esse valor de λ, por meio de uma simples inspeção. Sendo assim, uma vez que utilizamos os Algoritmo Segmento FEI para caminhar na curva que estar por baixo, e esse não precisa saber a *priori* o valor de λ na extremidade direita da FEI, podemos começar diretamente de $\lambda \to +\infty$. O único inconveniente dessa abordagem é que, como o Algoritmo das Interseções precisa da coordenada de dois pontos para determinar a interseção de duas parábolas, e $\lambda \to +\infty$ não é um valor numérico calculável, fica impossível determiná-la de forma exata quando não sabemos o valor de λ_{max}. Entretanto, esse inconveniente pode ser facilmente contornado pelo fato das curvas associadas a carteiras distintas se comportarem como retas, ao invés de parábolas, para $\lambda > \lambda_{max}$. Na Subseção 6.8.2, explicaremos como contornar esse inconveniente.

6.4 Iniciando o método

No início do Algoritmo PFV, sabemos qual é a solução do Problema (3.1) para $\lambda \to +\infty$. Essa ideia pode ser estendida para todo o método, de forma que, no início de cada iteração, partimos de λ_1, o ponto mais à esquerda da parte já conhecida da fronteira eficiente do problema. A solução de (3.1) nessa coordenada, é denotada por x^1. Além desses dados, temos também as variáveis não nulas referentes à curva que está por baixo, que denotamos por C_1, assim como o conjunto ativo fixado e o valor da função objetivo de (3.1) nesse ponto. A Figura 6.2 ilustra a situação.

O passo inicial de uma iteração qualquer do algoritmo consiste em determinar o intervalo de λ a ser investigado, que é aquele aquele em que o conjunto ativo fixado da curva C_1 é constante. Na subseção a seguir, explicaremos em detalhes a forma como esse intervalo é determinado.

6.4.1 Determinando o intervalo a ser investigado

De posse das informações do ponto inicial, sabemos quais são os ativos que compõem a carteira referente a C_1, bem como quais são as restrições ativas fixadas nessa parte da curva.

Logo, podemos caminhar sobre a curva C_1, da direita para a esquerda, diminuindo o valor de λ, por meio do Algoritmo Segmento FEI-DE. Nesse caso, vamos percorrer um segmento de parábola, que denotaremos por s_1, chegando a um ponto de mudança de conjunto ativo, ou de parábola, que chamaremos de *ponto de quebra* de segmento, associado a um novo valor λ_2 e a uma solução $x^{2'}$. A Figura 6.3 ilustra essa situação.

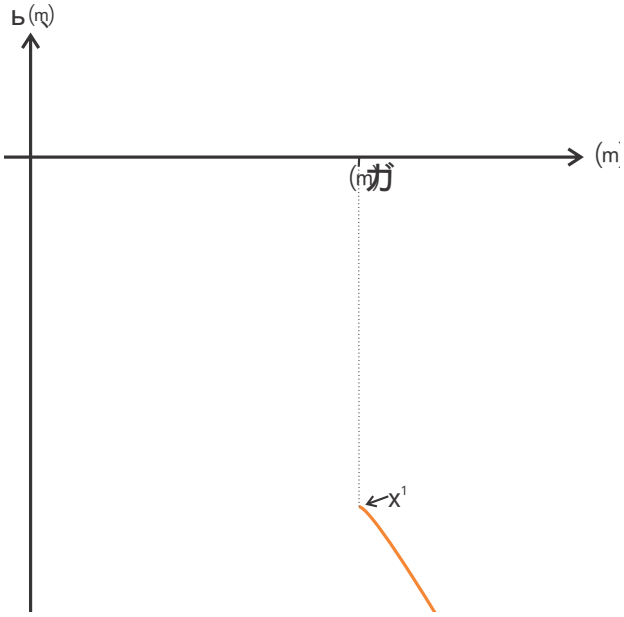


Figura 6.2: Situação inicial.

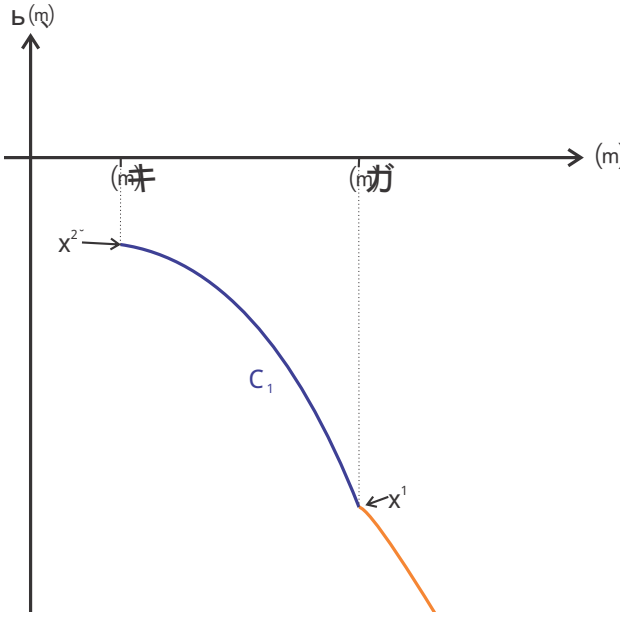


Figura 6.3: Intervalo Inicial.

Após a determinação do intervalo, o passo seguinte consiste em determinar se a curva C_1 será sempre aquela que está por baixo em $I_0 = [\lambda_2, \lambda_1]$. Na subseção a seguir, explicaremos em detalhes a forma como é feito esse teste.

6.4.2 Testando se C_1 é a curva que está por baixo

O primeiro passo desse teste consiste em determinar se a curva que está por baixo, quando $\lambda = \lambda_2$, é C_1. Para tanto, resolvemos o Problema (3.1) com esse valor de λ fixado, por meio do Algoritmo PQIM, e determinamos a solução do mesmo, que denotamos por x^2. Naturalmente, essa solução está relacionada a uma curva C_2, atrelada às variáveis binárias não nulas de x^2.

Caso $x^2 = x^{2'}$, concluímos que $C_2 = C_1$, e o segmento da fronteira eficiente do Problema (3.1) associado ao intervalo I_0 começa e acaba em C_1. Nesse caso, a curva que está por baixo nesse intervalo será sempre C_1, a menos que uma outra curva corte C_1 duas vezes no intervalo. A Figura 6.4 ilustra essa possibilidade.

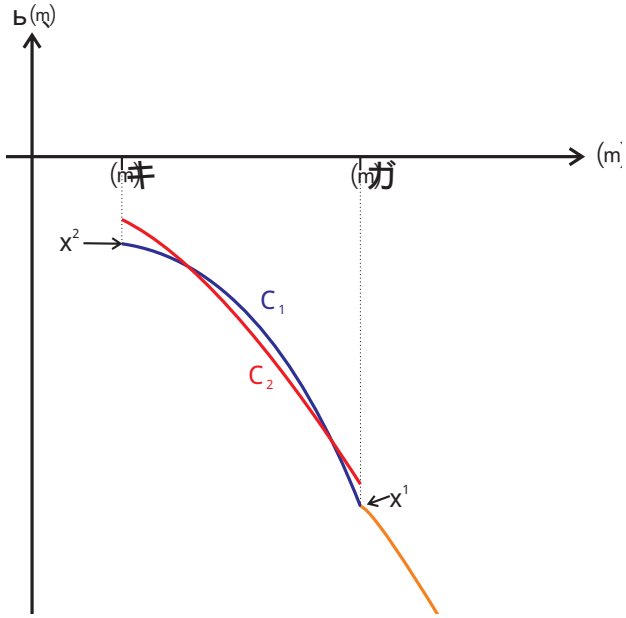


Figura 6.4: Exemplo de duas interseções.

Inicialmente, o Algoritmo PFV desconsidera esse caso, de forma que, se $x^2 = x^{2'}$, a parte da FEI do Problema (3.1) associada ao intervalo I_0 é o segmento de parábola s_1, da curva C_1. Entretanto, opcionalmente, é possível aplicar uma heurística para tentar encontrar as duas interseções, caso elas ocorram de fato. Na Seção 6.10 explicaremos uma maneira de considerar essa possibilidade. A Figura 6.5 ilustra o segmento da FEI do Problema (3.1) caso $x^2 = x^{2'}$.

Entretanto, nem sempre a curva que está por baixo no início e no final do intervalo I_0 é a mesma. Caso $x^2 \neq x^{2'}$, só podemos afirmar que, no início do intervalo I_0, a curva que está por baixo é C_2 e, no final do intervalo, a curva que está por baixo é C_1, conforme ilustramos na Figura 6.6.

Na seção a seguir, vamos mostrar como trabalhar com essa possibilidade.

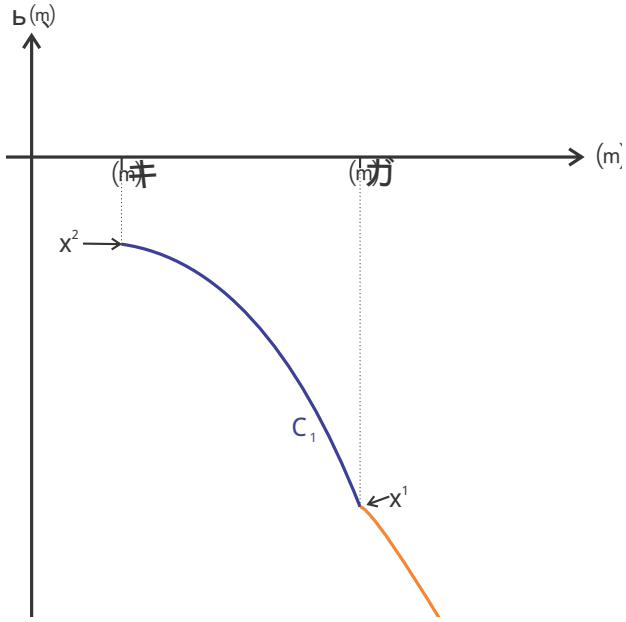


Figura 6.5: Fronteira eficiente do caso $x^2 = x^{2'}$.

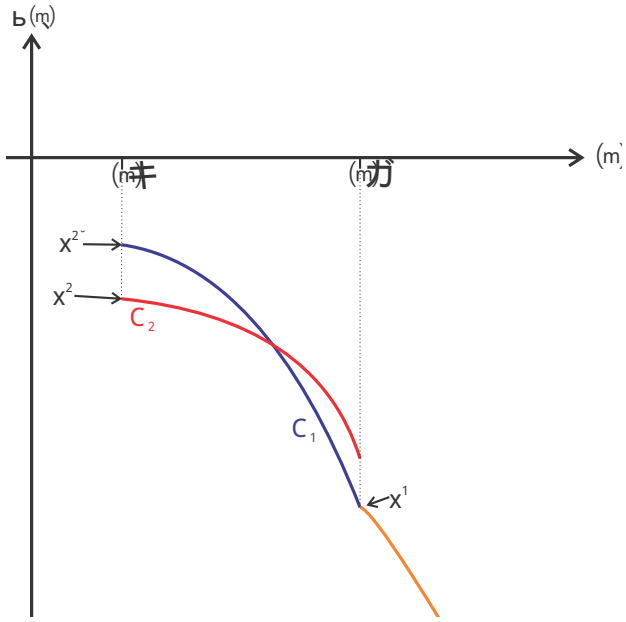


Figura 6.6: Caso $x^2 \neq x^{2'}$.

6.5 Analisando a interseção de duas curvas

Caso a curva mais abaixo em λ_1 seja C_1 e a curva mais abaixo em λ_2 seja C_2, podemos garantir que essas curvas se interceptam no interior do intervalo I_0, pois ambas as curvas são contínuas

e não crescentes. Logo, é preciso encontrar a interseção dessas curvas, para assim determinar os intervalos de λ em que cada uma delas está por baixo das demais.

Para encontrar essa interseção, precisamos determinar os segmentos de C_1 e C_2 associados ao intervalo I_0. Por construção, conhecemos o segmento da curva C_1, s_1, associado a esse intervalo, assim como o ponto da curva C_2 associado a λ_2. Para determinar o segmento da curva C_2, associado a esse intervalo, que denotaremos por s_2, determinamos o conjunto ativo fixo referente à curva C_2 em λ_2, e caminhamos, da esquerda para a direita no(s) segmentos(s) parábola(s) que compõem a curva C_2 no intervalo I_0.

Note que, a curva C_2 pode conter mais de um segmento de parábola atrelado ao intervalo I_0, cada um correspondente a um conjunto ativo fixo. Para obtê-los, aplicamos o Algoritmo Segmento FEI-ED, partindo de $\lambda = \lambda_2$, repetidamente, até que a última "quebra" de segmento ocorra para um valor de λ maior ou igual a λ_1. Dessa forma, o trecho da curva C_2, referente ao intervalo I_0, será composto pela união de todos os segmento de parábola existentes nesse intervalo. Caso tenhamos k segmentos distintos e, denotando-os por s_2^1, s_2^2,...,s_2^k, podemos afirmar que $s_2 = s_2^1 \cup s_2^2 \cup ... \cup s_2^k$.

A interseção de C_1 e C_2 no intervalo I_0 pode ser determinada pela coordenada, λ, em que f_1 e f_2 possuem o mesmo valor. Para encontrar essa coordenada, parametrizamos esses dois segmentos em função de λ, por meio de (6.2), e encontramos, pelo Algoritmo das Interseções, a coordenada de encontro deles, que denotamos por λ_3. Note que, como s_2 é composta pela união dos segmentos de parábola s_2^1, s_2^2,...,s_2^k, devemos determinar qual dos k segmentos de s_2 se encontra com s_1. Como não sabemos a priori em qual dos segmentos essa interseção ocorre, testamos, por meio do Algoritmo das Interseções, a possibilidade de interseção entre s_1 e cada um dos k segmentos que compõem s_2.

A partir do valor de λ_3, é possível encontrar a solução do subproblema do tipo (3.2) referente à coordenada de interseção de C_1 e C_2. Como cada uma das curvas se refere a uma carteira diferente, é natural que a solução obtida para cada curva não seja a mesma. Porém, o valor ótimo da função objetivo de ambos os subproblemas será o mesmo em λ_3. Denotaremos essas soluções por $x^{3'}$ e $x^{3''}$. A Figura 6.7 ilustra essa interseção.

De posse da coordenada λ_3, podemos encontrar a curva que está por baixo na interseção de C_1 e C_2. Para encontrá-la, resolvemos, pelo Algoritmo PQIM, o Problema (3.1) com $\lambda = \lambda_3$ fixado, determinando uma solução, que denotamos por x^3, cujos índices das variáveis binárias não nulas estão atrelados a uma curva C_3.

Caso $x^3 = x^{3'}$ ou $x^3 = x^{3''}$, não existe nenhuma curva abaixo da interseção de C_1 e C_2, ou seja, o ponto da FEI do Problema(3.1) que está por baixo em $\lambda = \lambda_3$, pertence a ambas as curvas. Logo, excluindo a existência de duas interseções entre C_1 e C_2, a curva que está por baixo em

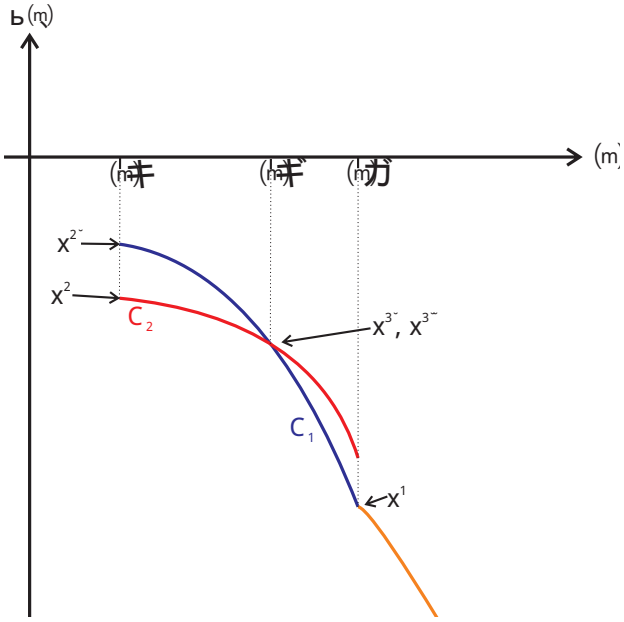


Figura 6.7: Encontrando a interseção das curvas.

$[\lambda_2,\lambda_3]$ é C_2 e a curva que está por baixo em $[\lambda_3,\lambda_1]$ é C_1, ou seja, a curva que está por baixo em I_0, é a união do segmento de C_2, que está no intervalo $[\lambda_2,\lambda_3]$, com o segmento de C_1, que está no intervalo $[\lambda_3,\lambda_1]$. A Figura 6.8 ilustra essa situação. Já a Figura 6.9 ilustra a FEI em I_0, formada pelos segmentos que estão abaixo dos demais.

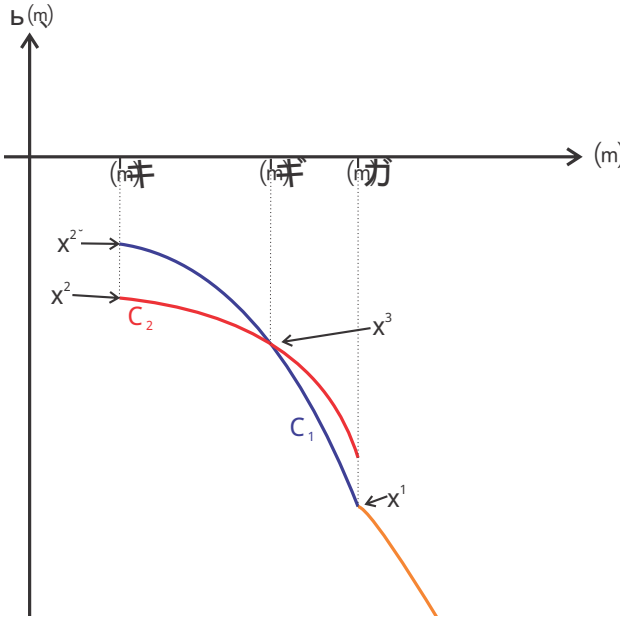


Figura 6.8: Caso $x^3 = x^{3'}$ ou $x^3 = x^{3''}$.

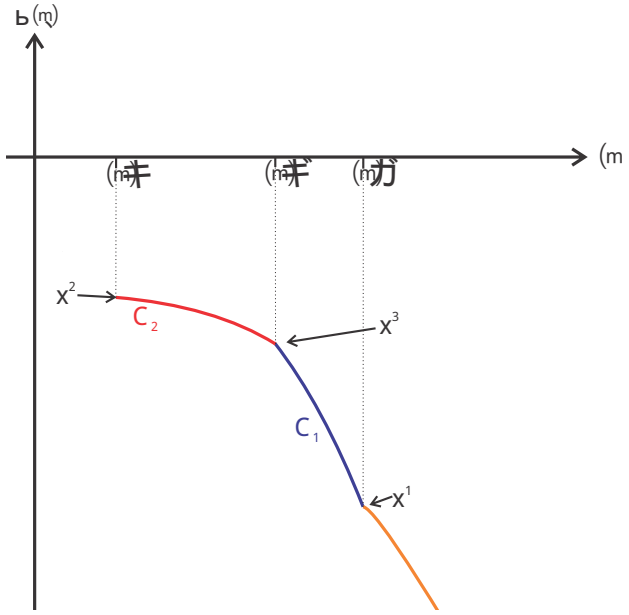


Figura 6.9: FEI do caso $x^3 = x^{3'}$ ou $x^3 = x^{3''}$.

Entretanto, é possível que exista uma curva abaixo da interseção de outras duas, ou seja, pode ocorrer que $x^3 \neq x^{3'}$ e $x^3 \neq x^{3''}$. Nesse caso, podemos apenas afirmar que, no início do intervalo $[\lambda_2,\lambda_3]$, a curva que está por baixo é C_2 e, no final, a curva que está por baixo é C_3. Da mesma forma, no início do intervalo $[\lambda_3,\lambda_1]$, a curva que está por baixo é C_3 e, no final, a curva que está por baixo é C_1. Essa situação sugere que é possível dividir o problema de encontrar o segmento da FEI no intervalo I_0 em dois subproblemas definidos em intervalos complementares. A Figura 6.10 ilustra essa possibilidade.

Na seção a seguir, vamos mostrar como trabalhar com esse caso.

6.6 Particionando o problema

Da análise feita nas duas seções anteriores, é possível que a FEI de (3.1), no intervalo I_0, tenha que ser determinada por meio de dois subproblemas em intervalos complementares $[\lambda_2,\lambda_3]$ e $[\lambda_3,\lambda_1]$. Note ainda que, por construção, cada um dos dois subproblemas é análogo ao problema original, de modo que o procedimento sugerido na Seção 6.5 pode ser aplicado a ambos. As Figuras 6.11, 6.12 e 6.13 mostram o processo de subdivisão do problema original em dois subproblemas.

É fácil perceber que, a exemplo do problema original, a aplicação do processo descrito na Seção 6.5 sobre um dos subproblemas, irá resolvê-lo, obtendo uma parte da FEI, ou gerar mais dois subproblemas. Esse processo, na pior das hipóteses, pode gerar uma árvore de subproblemas.

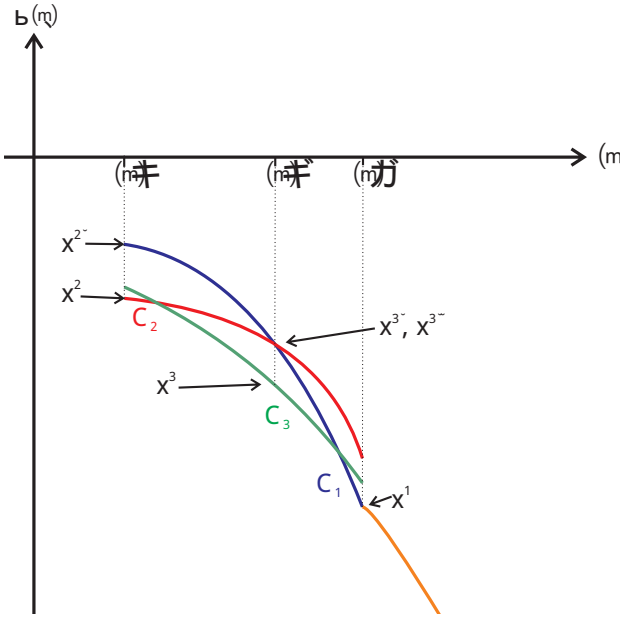


Figura 6.10: Subproblemas complementares.

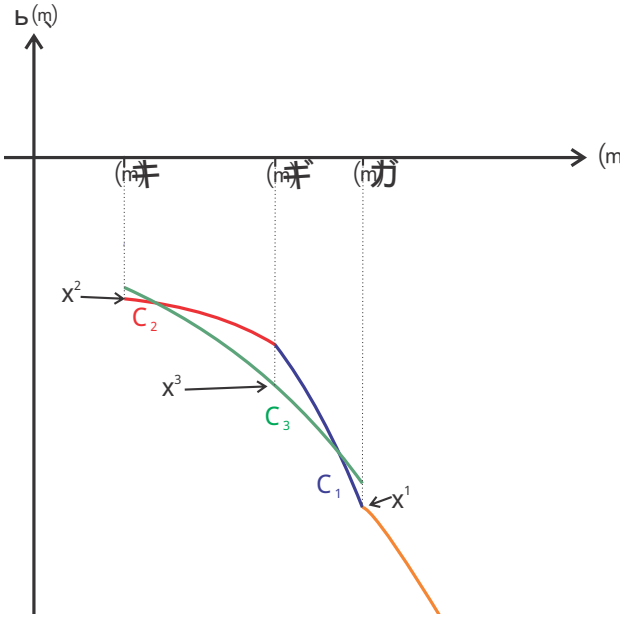


Figura 6.11: Subproblemas complementares 2.

Entretanto, como existe um número finito de curvas distintas que podem estar por baixo em cada subproblema, o número máximo de partições que podem ser feitas em uma iteração também o será, de forma que teremos no máximo $T-1$ delas, onde T é o número de carteiras distintas associadas ao Problema (3.1). Portanto, a resolução da árvore gerada por cada subproblema possui terminação

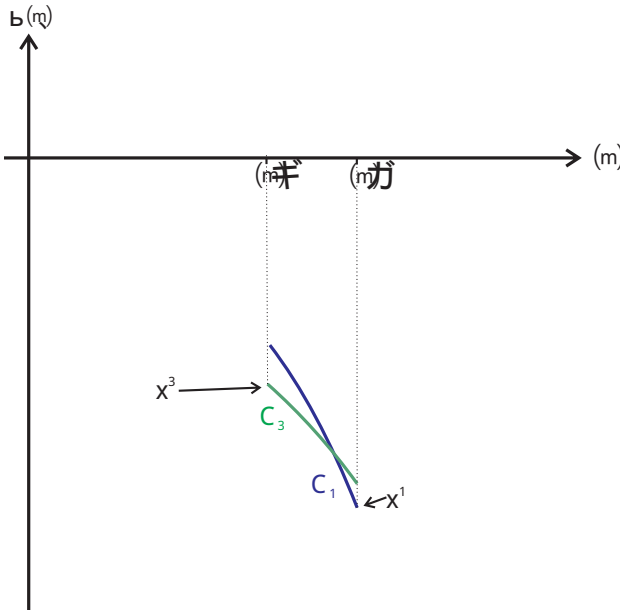


Figura 6.12: Subproblema 1.

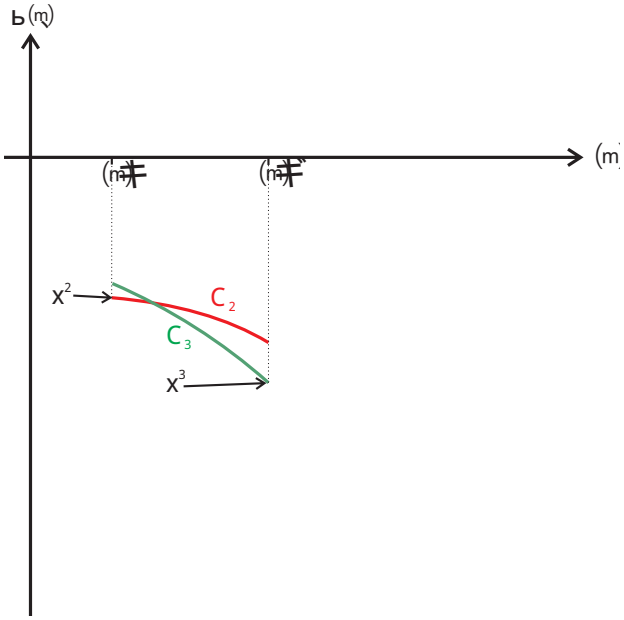


Figura 6.13: Subproblema 2.

finita, de forma que a iteração do Algoritmo PFV acaba quando conseguimos resolver todos os subproblemas e assim achar a curva que está por baixo no intervalo $[\lambda_2,\lambda_1]$

Para facilitar a compreensão do texto, denotaremos por *subproblema complementar*, cada um dos subproblemas gerados. A Figura 6.14 representa o segmento da fronteira eficiente correspon-

dente à nossa iteração, caso os dois subproblemas complementares tenham solução imediata.

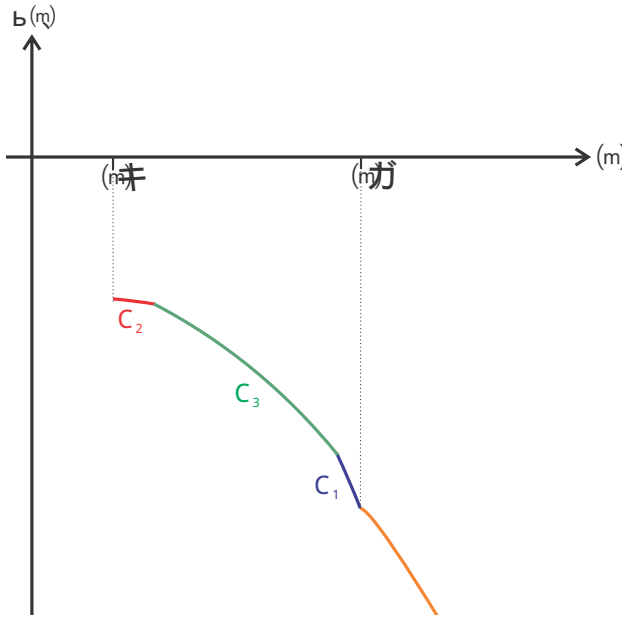


Figura 6.14: FEI do caso em que os dois subproblemas possuem solução imediata.

6.6.1 Um algoritmo para o subproblema complementar

Cada subproblema está associado a um intervalo $[\lambda_{k+1},\lambda_k]$, tendo como informação inicial o índice das variáveis que compõem a curva que está por baixo em λ_{k+1}, que denotaremos por C_j, assim como o índice das variáveis que compõem a curva que está por baixo em λ_k, que denotaremos por C_i. Além disso, também são conhecidas as soluções de (3.1) nos extremos do intervalo, x^{k+1} e x^k. Para facilitar a compreensão do texto, chamaremos de $SPC(\lambda_i,\lambda_f)$, o subproblema referente ao intervalo $[\lambda_i,\lambda_f]$.

Baseados nas ideias descritas nas duas seções anteriores, implementamos um algoritmo que resolve um subproblema complementar ou gera mais dois deles. Seu pseudocódigo é apresentado no Algoritmo 8.

Em seu passo inicial, determinamos a união de segmentos de parábola determinados por C_i e C_j no intervalo $[\lambda_{k+1},\lambda_k]$ (linhas 1 e 2). Depois, calculamos a interseção desses segmentos (linha 3), assim como a curva que está por baixo na coordenada de encontro (linha 4). Caso não exista uma curva abaixo dessa interseção, determinamos, na linha 6, a solução imediata do subproblema complementar, senão geramos dois novos problemas (linha 8).

Algoritmo 8: Um algoritmo para resolver o SPC(λ_{k+1},λ_k)

1 Partindo de λ_k, determine, pela aplicação repetida do Algoritmo Segmento FEI-DE, os segmentos de parábola s_i^1, s_i^2,...,$s_i^{k_1}$ associados ao intervalo $[\lambda_{k+1},\lambda_k]$ da curva C_i, onde k_1 é o índice do último segmento encontrado;
2 Partindo de λ_{k+1}, determine, pela aplicação repetida do Algoritmo Segmento FEI-ED, os segmentos de parábola s_j^1, s_j^2,...,$s_j^{k_2}$ associados ao intervalo $[\lambda_{k+1},\lambda_k]$ da curva C_j, onde k_2 é o índice do último segmento encontrado;
3 Encontre, por meio do Algoritmo das Interseções, a interseção $(\overline{\lambda},\overline{f})$ dos segmentos s_i^1, s_i^2,...,$s_i^{k_1}$ de C_i com os segmentos s_j^1, s_j^2,...,$s_j^{k_2}$ de C_j;
4 Por meio do Algoritmo PQIM, calcule x^* e f^*, referentes à solução de (3.1) com $\lambda = \overline{\lambda}$;
5 **se** $\left|\overline{f} - f^*\right| \leq 10^{-6}$, **então**
6 | O segmento referente ao subproblema é a união do trecho de C_j que está em $[\lambda_{k+1},\overline{\lambda}]$ com o trecho de C_i que está em $[\overline{\lambda},\lambda_k]$;
7 **senão**
8 | Gere os subproblemas complementares SPC(λ_{k+1},$\overline{\lambda}$) e SPC($\overline{\lambda}$,λ_k);
9 **fim**

6.7 Trabalhando com a árvore de subproblemas

O Algoritmo PFV, a cada iteração, precisa guardar os pontos chave da FEI, que são tanto aqueles em que ocorre mudança de carteira, quanto aqueles em que há uma mudança de segmento de parábola de uma curva fixada. Em geral, as informações relevantes para um ponto de abscissa λ_i são: os índices das variáveis binárias referentes à curva que está por baixo, a solução de (3.1) para esse valor de λ fixo, e o valor da função objetivo $f(\lambda_i)$.

Como essas informações sobre os pontos chave são constantemente utilizadas, é necessário criar um vetor de estruturas para armazená-las de forma adequada. Denotando por **fe** esse vetor, temos:

- **fe(i).λ** armazena o valor λ_i, referente a um ponto chave da FEI..
- **fe(i).vb** armazena o índice das variáveis binárias não nulas refente à curva que está por baixo em λ_i.
- **fe(i).x** armazena a solução do Problema (3.1), x^i, referente à curva que está por baixo em λ_i.
- **fe(i).f** armazena o valor da função objetivo avaliada em $\lambda = \lambda_i$ e $x = x^i$.

Esse vetor de estruturas, além de conter dados de cada ponto chave da FEI, também pode ser utilizado para guardar as informações dos extremos dos subproblemas complementares, apresen-

tados na seção anterior. Seguindo essa ideia, decidimos trabalhar com dois vetores de estruturas, um que armazena a informação dos extremos ótimos determinados pelo Algoritmo PFV e outro que armazena as informações temporárias dos extremos dos subproblemas complementares ainda não analisados. Denotaremos o segundo vetor por **temp**.

Em geral, o vetor **fe** caracteriza o extremo da direita, que já é ótimo, e o vetor **temp** caracteriza o extremo da esquerda, que ainda necessita ser investigado. Devido a essa caracterização particular, é possível trabalhar com os subproblemas complementares da mesma forma que trabalhamos com o problema principal, i.e, os subproblemas são investigados da direita para a esquerda. Na subseção a seguir, explicaremos com detalhes essa ideia.

6.7.1 Investigando a árvore de subproblemas da direita para a esquerda

Como o Algoritmo PFV trabalha da direita para a esquerda, o subproblema a ser analisado, em uma sub-iteração qualquer, tem como extremo direito o ponto mais à esquerda da FEI já conhecido, e como extremo esquerdo o ponto mais à direita dentre aqueles que ainda precisam ser investigados. Essa estrutura particular de nosso problema sugere que os pontos chave da FEI devam ser sempre empilhados em **fe**, ao passo que os extremos não analisados devem ser empilhados em **temp**, até o momento em que eles passam a ser de fato ótimos, sendo assim transferidos para o topo da pilha de **fe**.

Caso o problema referente a uma iteração qualquer do Algoritmo PFV não tenha solução imediata, precisamos trabalhar, de forma iterativa, com os subproblemas complementares. Em uma sub-iteração qualquer, o subproblema complementar a ser analisado será sempre aquele em que o extremo direito é representado pelo topo da pilha de **fe** e o extremo esquerdo será sempre representado pelo topo da pilha de **temp**. Nessa sub-iteração, atualizamos as duas estruturas da seguinte maneira:

- Caso o subproblema complementar não possua solução imediata, podemos dividi-lo em outros dois, de modo que o mais à direita seja aquele a ser analisado na próxima sub-iteração. Para que isso ocorra, uma vez que o extremo esquerdo desse subproblema é o ponto de interseção das duas curvas analisadas e o extremo direito não muda, basta colocarmos essa interseção no topo da pilha de **temp**.

- Caso o subproblema complementar possua solução imediata, incluímos em **fe** as informações da interseção das duas curvas e do extremo esquerdo do intervalo que define o subproblema, além de excluirmos de **temp** o extremo da sub-iteração atual, pois ele já é ótimo.

Note que, no primeiro caso, o subproblema complementar é dividido em dois outros, de forma que, na próxima sub-iteração, continuaremos trabalhando com o mesmo subproblema, ainda da direita para a esquerda, só que agora de forma dividida. Já no segundo caso, o subproblema complementar é resolvido, e o segmento da FEI de (3.1) referente ao mesmo é armazenado em **fe**. Nessa situação, o subproblema complementar a ser resolvido na próxima sub-iteração é adjacente àquele resolvido na sub-iteração atual. Seguindo esse raciocínio, as sub-iterações terminam quando incluímos em **fe** o extremo esquerdo do problema original. Da mesma forma, a iteração principal do Algoritmo PFV termina quando o vetor de estruturas **temp** estiver vazio.

Logo, após o final de uma iteração do método, começamos a próxima iteração com o menor valor de λ para o qual conhecemos a solução do Problema (3.1), ou seja, partimos de λ_2. Essa rotina é repetida até o momento em que encontramos a extremidade esquerda da FEI, ou seja, quando $\lambda = 0$. A partir desse processo, o método encontra uma aproximação da fronteira eficiente do nosso problema na qual desconsiderarmos a possibilidade de duas curvas distintas se cruzarem duas vezes em uma mesma iteração. A Figura 6.15 ilustra uma possibilidade para a fronteira eficiente final do nosso problema.

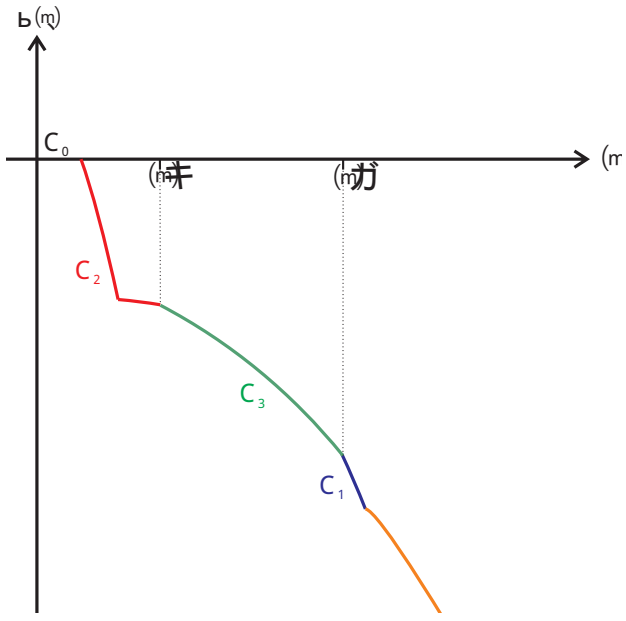


Figura 6.15: Fronteira eficiente final.

6.8 Encontrando a interseção nas extremidades

Uma das maiores dificuldades encontradas pelo Algoritmo PFV é determinar a interseção de duas curvas nas proximidades dos extremos da fronteira eficiente de (3.1), ou seja, encontrar o ponto de mudança referente à curva associada ao ativo de maior retorno ($\lambda \to +\infty$) e à curva nula, que representa o risco mínimo ($\lambda \to 0$). Nas subseções a seguir, explicaremos o motivo dessas dificuldades, assim como a forma que elas foram tratadas.

6.8.1 A interseção de duas curvas quando $\lambda \to 0$

Conforme discutimos na Seção 6.3, para $\lambda \to 0$, o Problema (3.1) possui solução nula, que por sua vez é singular. Tal singularidade faz com que a curva associada a essa solução, que denotaremos por C_0, seja a reta constante $f(\lambda) = 0$, que não possui ponto de quebra de segmento, pois o conjunto ativo nunca muda. Sendo assim, o Algoritmo Segmento FEI não é aplicável a essa curva. Essa impossibilidade faz com que o Algoritmo das Interseções seja incapaz de encontrar o intercepto do segmento nulo com a curva adjacente, já que o Algoritmo Segmento FEI é usado para determiná-lo.

Como a imagem de C_0 é zero para qualquer valor de λ, podemos afirmar que ela está por baixo dos segmentos de curvas que apresentam um valor positivo de $f(\lambda)$. Como o risco nunca é negativo, vai existir um intervalo de λ em que todas as curvas, com exceção de C_0, estão acima do eixo λ, ou seja, a curva que está por baixo certamente será a curva nula. Tomando como referencial o sentido da direita para a esquerda, encontrar a curva cujo segmento é adjacente à C_0 na FEI do Problema (3.1), é equivalente a determinar qual das $T-1$ curvas restantes cruza por último o eixo λ na coordenada final, que chamaremos de λ_0. Explicaremos esse processo por meio da Figura 6.16, que ilustra essa situação.

Começando por λ_1, para determinarmos se a curva C_2 cruza C_0 na FEI de nosso problema, primeiramente encontramos o intervalo I_0 a ser trabalhado da forma usual, e depois testamos se $f(\lambda_2) > 0$. Como isso ocorre, parametrizamos a curva C_2 em função de λ e encontramos a interseção da mesma com o eixo das abscissas, λ_c, por uma versão modificada do Algoritmo das Interseções. Logo após, determinamos, por meio do Algoritmo PQIM, a curva que está por baixo em λ_c, que nesse caso é a curva C_1. Sendo assim, C_2 não é a última curva a cruzar C_0, de forma que temos uma iteração usual do Algoritmo PFV sobre o intervalo $[\lambda_c,\lambda_1]$.

Começando agora de λ_c, determinamos que o próximo ponto de quebra, que ocorre em $\lambda = 0$, está acima do eixo λ e encontramos a interseção de C_1 com esse eixo, que é λ_0. De forma análoga, determinamos a curva que está abaixo de C_1 em λ_0, que nesse caso é a C_0, fazendo com que C_1 seja

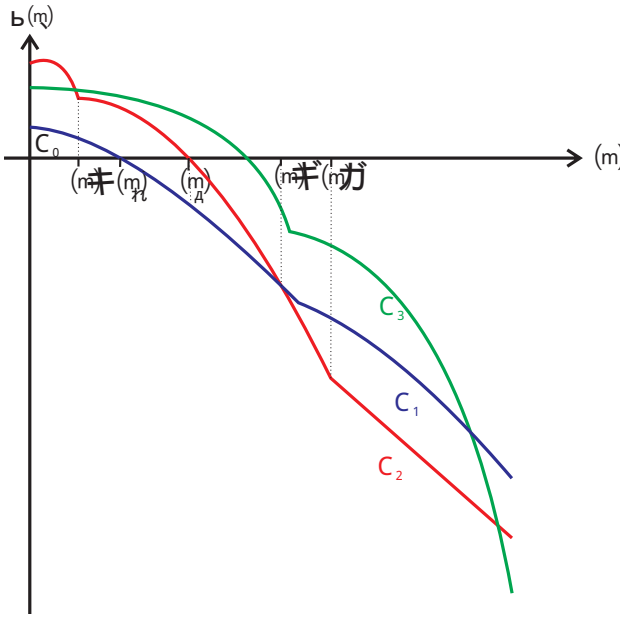


Figura 6.16: Fronteira perto da origem.

a última curva a cruzar o eixo λ. Sendo assim, podemos afirmar, por meio do referencial teórico já apresentado, que a curva que está por baixo no intervalo $[0,\lambda_0]$ é C_0, enquanto a curva que está por baixo no intervalo $[\lambda_0,\lambda_3]$ é C_1.

6.8.2 A interseção de duas curvas quando $\lambda \to +\infty$

Conforme discutimos na Seção 6.3, a solução de (3.1) para $\lambda \geq \lambda_{max}$ não muda, ou seja, $f(\lambda)$ é um reta decrescente para valores grandes de λ. Isso também vale para todas as T curvas associadas à carteiras distintas de (3.1), ou seja, o segmento mais à direita é uma reta decrescente. Logo, determinar a interseção de duas curvas quando $\lambda \geq \lambda_{max}$, pode ser equivalente a encontrar a interseção de duas retas.

A parametrização de cada uma das retas é similar àquela apresentada em (6.2), com exceção do fato que o termo quadrático, a, é nulo. De posse dessa informação e lembrando que a interseção de duas retas não paralelas no plano é única, podemos encontrar a coordenada comum às duas retas, λ_{int}, ao calcularmos os coeficientes b e c de acordo com o Algoritmo das Interseções e atribuirmos $\lambda_{int} = \frac{-c}{b}$.

Uma vez que, para valores de lambda maiores que λ_{max} todas as curvas associadas a carteiras distintas passam a se comportar como retas, ao invés de parábolas, podemos utilizar essa abordagem para encontrar a interseção de duas curvas quando iniciamos o método com $\lambda \to +\infty$, no

lugar desse valor particular de λ, cuja determinação exata não está definida. A Figura 6.17 mostra uma situação em que temos que encontrar a interseção de duas curvas (λ_{int}) para $\lambda \geq \lambda_1$, tendo como ponto de partida as soluções de (3.1) em $\lambda = \lambda_1$ e $\lambda \to +\infty$, o que, em primeira instância, é impossível de se determinar, pois não podemos parametrizar as duas curvas com o ponto no infinito.

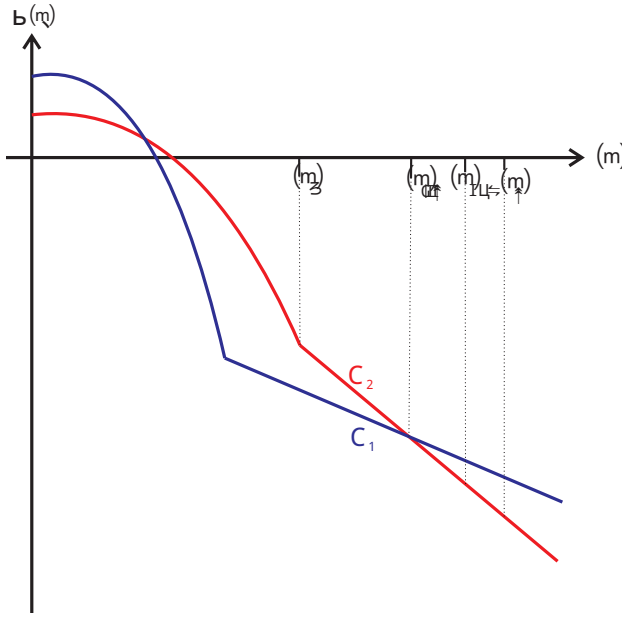


Figura 6.17: Fronteira perto do infinito.

Entretanto, note que, nessa faixa de valores de λ, todas as T curvas passam a ser retas, de forma que encontrar a interseção de duas curvas no infinito nada mais é do que encontrar a interseção de duas retas. Como uma reta é definida por dois pontos, sendo eles não necessariamente os de quebra, basta determinarmos um segundo ponto de cada reta - nesse caso, aqueles associados a λ_t - para podermos parametrizá-las e assim encontrarmos a interseção das mesmas. Essa determinação, por sua vez, pode ser feita sem maiores dificuldades, pois, como nessa faixa de valores a solução em x é sempre a de máximo retorno, a função objetivo será linear.

6.9 O nosso algoritmo

Por meio das ideias apresentadas nesse capítulo, desenvolvemos o Algoritmo PFV, que é a quinta, e, certamente, a mais importante de nossas contribuições. Seu pseudocódigo pode ser visto no Algoritmo 9.

Algoritmo 9: O Algoritmo PFV

Determine as informações da extremidade direita da FEI do Problema (3.1);
Faça $v \leftarrow 1$;
Guarde, em **fe(v)**, as informações referentes a $\lambda \rightarrow +\infty$;
enquanto $fe(v).f \leq -10^{-8}$ **faça**
    Partindo de **fe(v).**λ, determine, pelo Algoritmo Segmento FEI-DE, as informações $\widetilde{\lambda}$, $\widetilde{x}$ e $\widetilde{f}$ da extremidade final do segmento de parábola em que o conjunto ativo de **fe(v).x** é constante;
    se $\widetilde{f} > 10^{-8}$, **então**
        Encontre, pelo Algoritmo das Interseções, a coordenada $\underline{\lambda}$ de interseção do segmento de parábola, encontrado no passo anterior, com o eixo λ;
        Faça $\widetilde{\lambda} \leftarrow \underline{\lambda}$;
    fim
    Por meio do Algoritmo PQIM, calcule $\widehat{x}$ e $\widehat{f}$, referentes à solução de (3.1) com $\lambda = \widetilde{\lambda}$;
    se $\|\widetilde{x} - \widehat{x}\|_2 \leq 10^{-6}$, **então**
        Faça $v \leftarrow v + 1$;
        Guarde, em **fe(v)** as informações referentes a $\widetilde{\lambda}$;
        se $\left|\widehat{f}\right| \leq 10^{-8}$, **então**
            Faça $v \leftarrow v + 1$;
            Guarde, em **fe(v)** as informações referentes à solução nula ($\lambda = 0$);
        fim
    senão
        Faça $r \leftarrow 1$ e remova todas as informações contidas em **temp**;
        Guarde, em **temp(r)**, as informações referentes a $\widetilde{\lambda}$;
        enquanto $r \neq 0$ **faça**
            $\lambda_1 \leftarrow$**fe(v).**λ e $\lambda_2 \leftarrow$**temp(v).**λ;
            Defina C_1 e C_2 como as curva referentes a **fe(v).vb** e **temp(f).vb**, respectivamente;
            Partindo de λ_1, determine, pelo Algoritmo Segmento FEI-DE, a união de segmentos de parábola, s_1, da curva C_1, associada ao intervalo $[\lambda_2,\lambda_1]$;
            Partindo de λ_2, determine, pelo Algoritmo Segmento FEI-ED, a união de segmentos de parábola, s_2, da curva C_2, associada ao intervalo $[\lambda_2,\lambda_1]$;
            Encontre, por meio do Algoritmo das Interseções, a interseção, $(\overline{\lambda},\overline{f})$, de s_1 e s_2;
            Fazendo uso Algoritmo PQIM, calcule x^* e f^* da solução de (3.1) com $\lambda = \overline{\lambda}$;
            se $\left|\overline{f} - f^*\right| \leq 10^{-6}$, **então**
                Guarde, em **fe**, as informações dos pontos de quebra de C_1 em $[\overline{\lambda},\lambda_1]$;
                Faça $v \leftarrow v + k_1^{'}$, onde $k_1^{'}$ é o número de pontos incluídos no passo anterior;
                Guarde, em **fe**, as informações dos pontos de quebra de C_2 em $[\lambda_2,\overline{\lambda}]$;
                Faça $v \leftarrow v + k_2^{'}$, onde $k_2^{'}$ é o número de pontos incluídos no passo anterior;
                Faça $r \leftarrow r - 1$;
            senão
                Faça $r \leftarrow r + 1$;
                Guarde, em **temp(r)**, as informações referentes à curva que está por baixo em $\overline{\lambda}$;
            fim
        fim
    fim
fim

No passo inicial desse algoritmo, determinamos a extremidade direita da FEI do nosso problema e a incluímos na lista de pontos chave, **fe** (linhas 1 a 3). Na linha 4, começamos a parte iterativa do nosso método, que termina no momento em que encontramos a extremidade esquerda da FEI.

Em um passo qualquer, partindo do ponto mais à esquerda conhecido da FEI, cujas informações estão armazenadas no topo de lista **fe**, determinamos, por meio do Algoritmo Segmento FEI-DE, o ponto de quebra referente à curva que está por baixo nesse ponto, assim como o intervalo a ser investigado nessa iteração (linha 5), seguindo a teoria descrita na Seção 6.4. Logo após, testamos a possibilidade de cruzamento da curva analisada com eixo λ nesse intervalo (linha 6). Caso isso ocorra, deslocamos o final do intervalo investigado para a interseção encontrada, de acordo com a teoria desenvolvida na Subseção 6.8.1 (linhas 7 e 8). Na sequência, fazendo uso da teoria explicada na Seção 6.4.2, testamos, por meio do Algoritmo PQIM, se a curva que está por baixo no final do intervalo é a mesma do início (linha 11). Caso isso ocorra, sabemos que essa curva é ótima em toda a extensão, de forma que guardamos as informações de seu final na lista de pontos chave e partimos para a próxima iteração (linhas 12 e 13). Além disso, se o final do intervalo tiver sido atualizado de acordo com as linhas 7 e 8, sabemos, pela teoria da Subseção 6.8.1, que a curva ótima adjacente à atual é a curva $f(\lambda) = 0$, cujo final está compreendido na extremidade esquerda da FEI. De forma que, podemos incluir as informações referentes à curva $f(\lambda) = 0$ em nossa lista de pontos chave e terminar o algoritmo (linhas 15 e 16).

Na possibilidade da curva que está por baixo no final do intervalo ser diferente daquela do início, dividimos, por meio da teoria descrita na Seção 6.6, o problema atual em dois subproblemas complementares (linhas 19 e 20). Em seguida, iniciamos a parte recursiva do Algoritmo PFV, que consistem em, a cada iteração, resolver um subproblema complementar ou dividi-lo em outros dois (linha 21). Tal processo é repetido até o momento em que todos os subproblemas forem resolvidos, possibilitando-nos a determinar a FEI no intervalo de busca original e assim terminar a iteração principal do nosso método. Cada iteração dessa parte é análoga ao Algoritmo 8. A única diferença aparece nas linhas 35 e 36, em que utilizamos a lista de pontos chave temporária **temp** para guardar os subproblemas complementares, de acordo com o processo descrito na Seção 6.7.

6.10 Determinando as interseções de duas curvas distintas

Conforme mencionamos na Subseção 6.4.2, podemos determinar, por meio de uma heurística, se a curva que está por baixo nos extremos de um intervalo é ótima em todo a sua extensão. A Figura 6.18 ilustra um caso em que ela não é ótima, pois apesar de C_1 estar por baixo tanto em λ_1 quanto em λ_2, existe uma curva C_2 que a corta duas vezes nesse intervalo, estando assim por

baixo em um subintervalo de I_0.

Nessa situação, uma maneira de determinar se existe uma curva C_2 que corta C_1 duas vezes, seria discretizar o intervalo $[\lambda_2,\lambda_1]$ e determinar, por meio do Algoritmo PQIM, se existe uma curva por baixo, diferente de C_1, em cada um dos pontos de discretização. Caso não exista, assumimos que a curva que está por baixo nesse intervalo é sempre C_1, senão, sabemos que existe uma coordenada, λ_3, para a qual a curva que está por baixo no início do intervalo $[\lambda_2,\lambda_3]$ é C_2 e, no final, a curva que está por baixo é C_3. Da mesma forma, a curva que está por baixo no início do intervalo $[\lambda_3,\lambda_1]$ é C_3 e, no final, a curva que está por baixo é C_1. Logo, o problema de encontrar as duas interseções em I_0 pode ser dividido em dois subproblemas que buscam exatamente uma interseção e podem ser facilmente resolvidos pelo Algoritmo PFV.

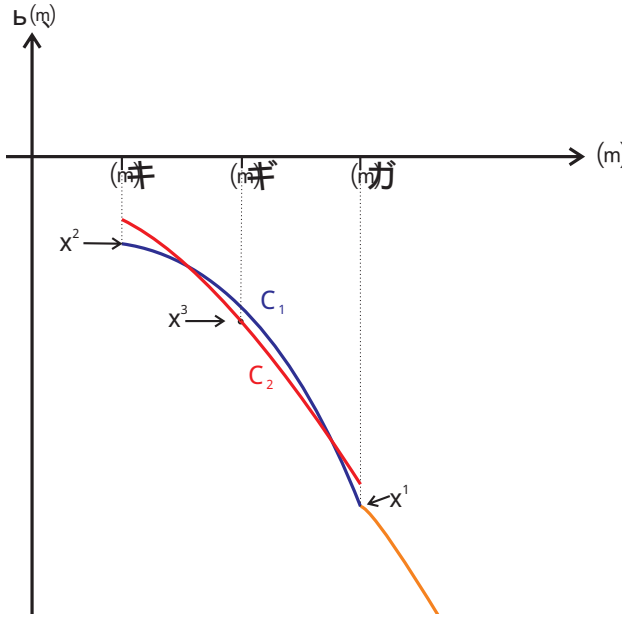


Figura 6.18: Subproblemas decorrentes das duas interseções.

Como a proposta inicial do nosso algoritmo procura não utilizar a técnica da discretização, pois ela é cara e não eficiente para o problema abordado, e não encontramos evidências concretas de que duas curvas se interceptam mais de uma vez na mesma iteração, resolvemos trabalhar com essa possibilidade de forma adaptativa, procurando essas interseções apenas em casos particulares. Nessa estratégia, testamos essa possibilidade apenas no ponto médio de um intervalo, desde que o comprimento do mesmo não seja muito pequeno (menor do que 0,1) e o extremo esquerdo dele não esteja muito próximo de zero.

Capítulo 7

Resultados computacionais

Neste capítulo, descrevemos os experimentos efetuados para investigar a eficiência do método que propusemos para aproximar a FEI do Problema (3.1). A análise do desempenho do algoritmo foi feita com base no tempo de execução. Além disso, comparamos o nosso método com aqueles, já difundidos na literatura, que encontram a aproximação da curva, tendo como critério tanto a fidelidade da aproximação da curva quanto o tempo computacional. Todos os testes foram feitos utilizando-se o programa MATLAB em um computador com processador AMD Phenon X4 965 64 bits 3,4GHZ e memória principal de 6 GB.

7.1 Os problemas testados

Para testar a eficiência do nosso programa, utilizamos o conjunto de problemas teste apresentados por Chang *et al.* [9]. Os cinco problemas selecionados correspondem a indicadores econômicos baseados em grandes bolsas de valores do mundo. A primeira instância, denominada Hang Seng, tem 31 ativos e está relacionada à bolsa de Hong Kong. A segunda, DAX, com 85 ativos, está associada à bolsa alemã. O terceiro problema, relativo ao mercado financeiro do Reino Unido, é denominado FTSE e tem 89 ativos. A quarta instância é a americana S&P, com 98 ativos. A última, denominada Nikkei, contém 225 ativos da bolsa japonesa[1]. Vale notar que as duas últimas instâncias são mais difíceis de se resolver numericamente, devido ao condicionamento de sua matriz de covariância.

Para cada instância, geramos, por meio do Algoritmo PFV, uma aproximação da FEI partindo de $\lambda_{max} = 10$. A escolha de um valor fixo para λ_{max} deve-se à necessidade de comparação do nosso método com aquele que envolve a discretização do intervalo $[0,\lambda_{max}]$, conforme veremos na

[1]Os dados estão disponíveis em `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html`.

Seção 7.4. Vale lembrar ainda que as matrizes de covariância relacionadas às cinco instâncias são simétricas e definidas positivas, de modo que a solução de cada subproblema é única. Restringimos a 10 o número de ativos que compõem a carteira (ou seja, tomamos $K = 10$) e utilizamos o limite mínimo de investimento de 5% para cada ativo (o que significa que $l_i = 0{,}05$, $i = 1, ..., n$).

7.2 Testando a eficiência do CHR

O primeiro teste feito consistiu em determinar a eficiência do método CHR no processo de obtenção da FEI de cada um dos cinco problemas. Nesse teste, para cada um dos problemas, calculamos o número de vezes que o CHR, combinado à heurística de melhoramento, foi capaz de resolver o Problema (3.1) com um valor de λ fixado, assim como o número de falhas que forçaram o uso do algoritmo *Branch-and-Bound* implícito. A Tabela 7.1 mostra os resultados obtidos.

Tabela 7.1: Teste de eficiência do CHR.

	Número de Chamadas	Número de acertos	Porcentagem de acerto
Hang Seng	15	7	46,7%
Dax	24	6	25,0%
FTSE	25	3	12,0%
S&P	17	4	23,5%
Nikkei	21	6	28,6%

A primeira coluna da tabela registra, para cada instância, o número de vezes que foi necessário resolver o Problema (3.1) por meio do Algoritmo PQIM. Já a segunda coluna mostra, para cada instância, o número de vezes que o Algoritmo PQIM foi capaz de resolver o Problema (3.1) sem o auxílio do Algoritmo BBI, ou seja, apenas pelo uso do Algoritmo CHR aliado a sua heurística de melhoramento. Finalmente, a terceira coluna representa, para cada instância, a porcentagem de vezes que o Algoritmo PQIM foi capaz de resolver os Problemas (3.1) somente com a combinação do Algoritmo CHR com a Heurística CHR (nessa seção, chamaremos essa junção de *Combinação CHR*).

É possível perceber, da Tabela 7.1, que a Combinação CHR não foi eficiente na resolução dos problemas inteiros associados à FEI dos cinco problemas de investimento escolhidos para teste. Vale também notar que essa combinação só conseguiu resolver os problemas do tipo (3.1) em que o valor de λ era relativamente grande, ou seja, ela foi incapaz de encontrar a solução inteira para

valores pequenos de λ, que são justamente aqueles que exigem maior tempo e precisão computacional. Ainda constatamos que, nesse último caso, o tempo necessário pela Combinação CHR para encontrar uma solução não inteira foi equivalente àquele gasto pelo algoritmo *Branch-and-Bound* implícito para encontrar a solução inteira sem o limitante superior obtido pela Heurística CHR.

Para contornar essa falta de eficiência, uma vez que o Algoritmo PFV sempre inicia na extremidade direita de FEI, onde os valores de λ são sempre grandes, criamos uma estratégia adaptativa para o Algoritmo PFV, que consiste em utilizar diretamente o algoritmo *Branch-and-Bound* implícito para resolver os Problemas (3.1), caso a Combinação CHR falhe mais de duas vezes durante o método geral.

7.3 As fronteiras eficientes obtidas

Nosso segundo teste consistiu em determinar graficamente a FEI alternativa de cada um dos 5 problemas no intervalo [0,10]. Além disso, uma vez que, para valores de λ próximos de zero a troca de carteira é constante e afeta a inclinação da FEI, optamos por também mostrar o comportamento da FEI de cada um dos problemas no intervalo [0;0,3]. As Figuras 7.1 a 7.10 mostram a FEI de cada um dos 5 problemas nesses dois intervalos particulares.

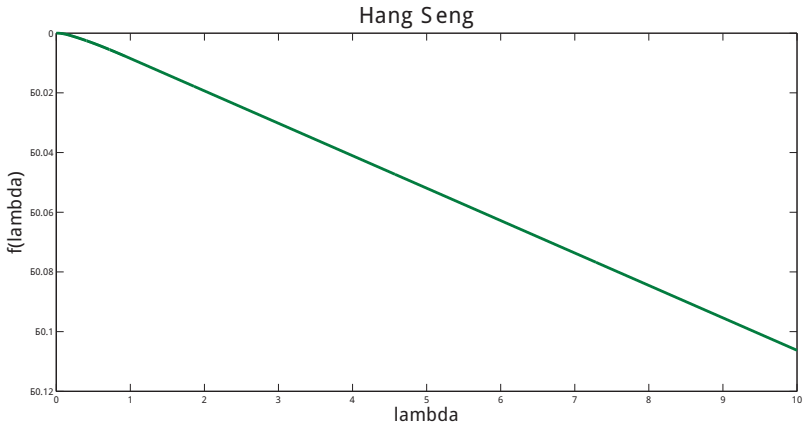


Figura 7.1: Instância Hang Seng com $\lambda \in [0,10]$.

É possível perceber que, em todos os casos, as fronteiras eficientes, quando traçadas no intervalo [0,10], não apresentam de forma clara tanto o aspecto quadrático, quanto as mudanças na inclinação e o intercepto com o eixo λ. Isso se deve ao fato de que essas mudanças começam a aparecer para valores pequenos de λ, em geral menores que 2, e a escala do gráfico não permite que elas sejam percebidas de forma imediata. Essa mudança fica mais evidente, em todos os problemas, quando

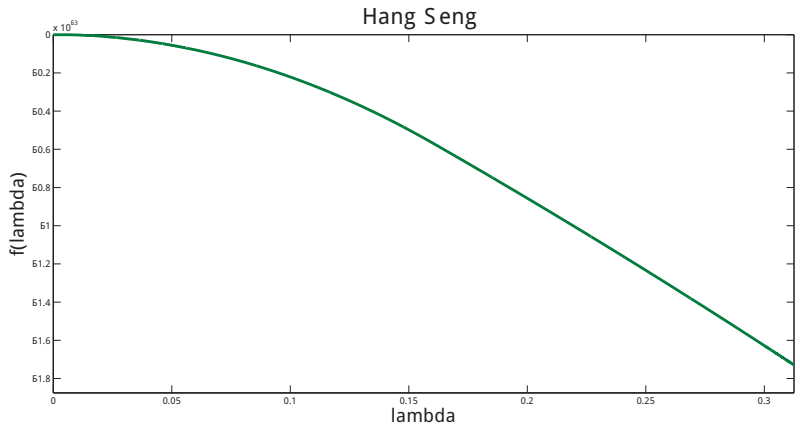


Figura 7.2: Instância Hang Seng com $\lambda \in [0; 0{,}3]$.

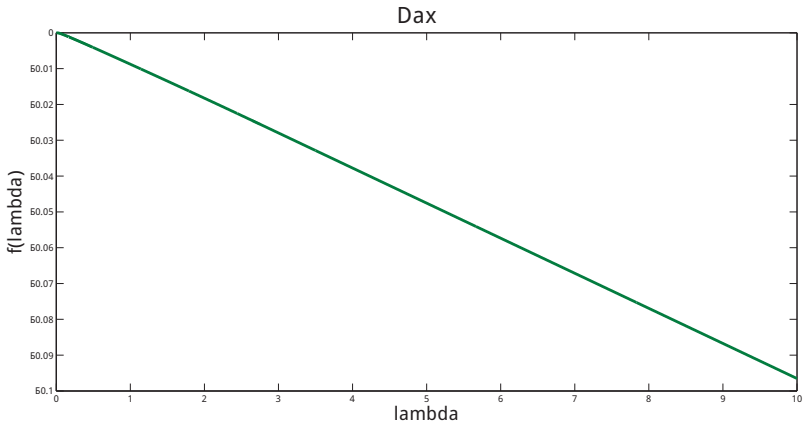


Figura 7.3: Instância Dax com $\lambda \in [0{,}10]$.

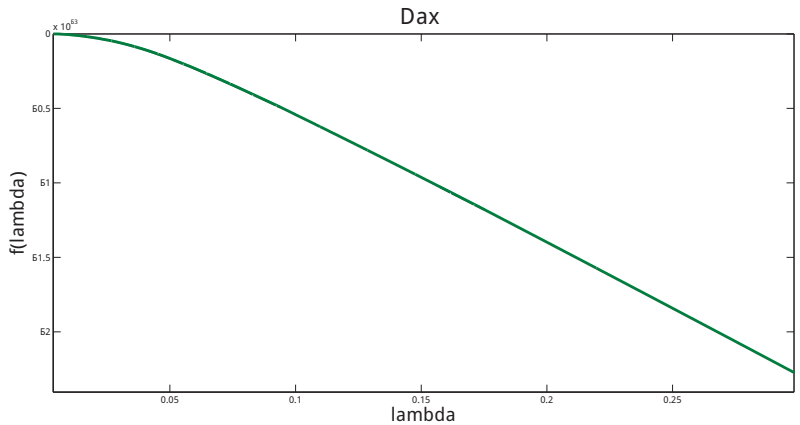


Figura 7.4: Instância Dax com $\lambda \in [0; 0{,}3]$.

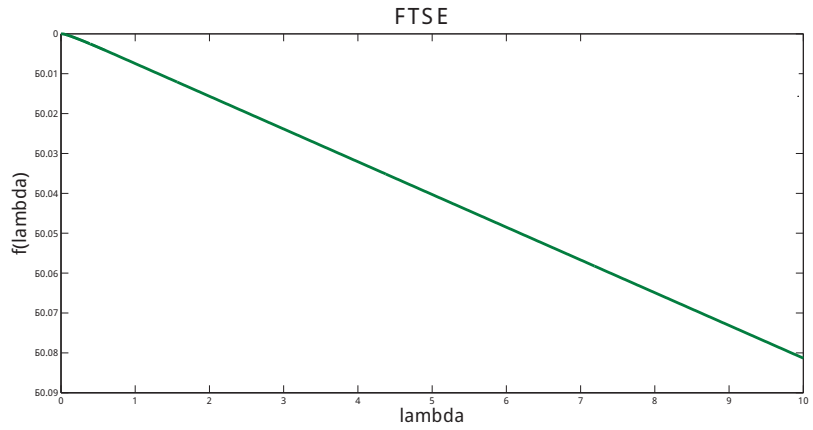


Figura 7.5: Instância FTSE com $\lambda \in [0{,}10]$.

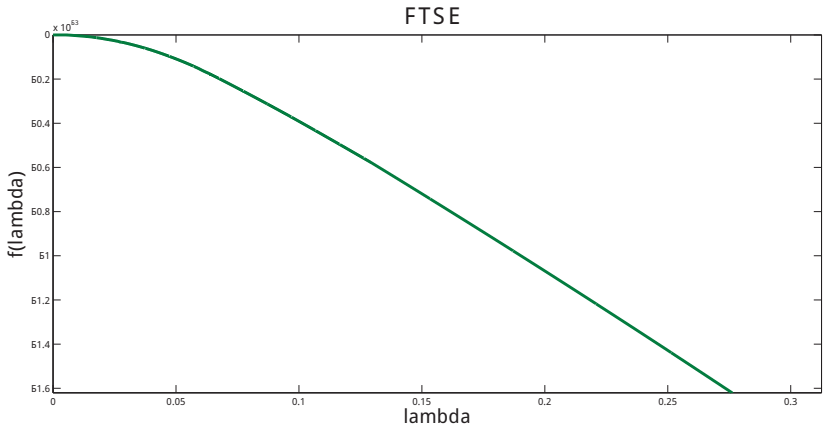


Figura 7.6: Instância FTSE com $\lambda \in [0; 0{,}3]$.

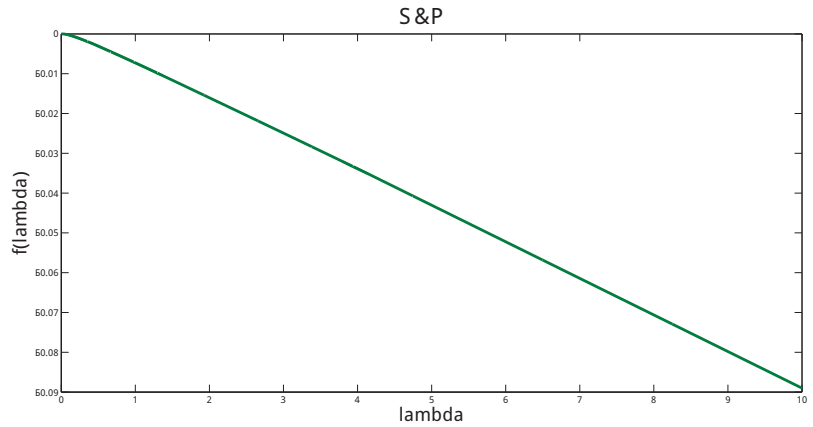


Figura 7.7: Instância S&P com $\lambda \in [0{,}10]$.

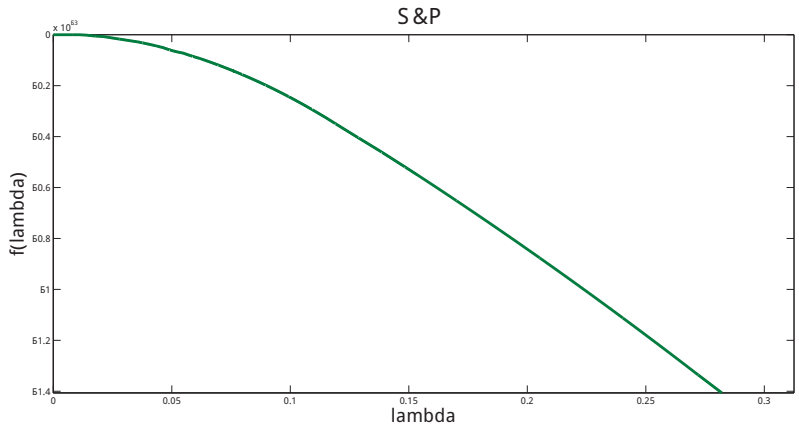


Figura 7.8: Instância S&P com $\lambda \in [0; 0{,}3]$.

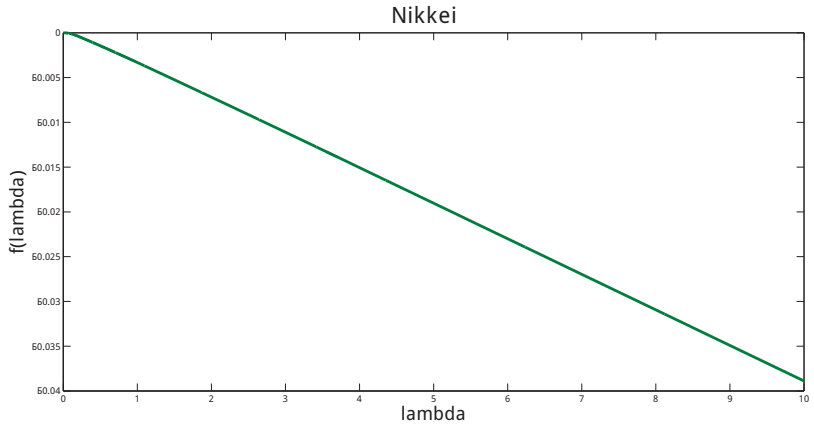


Figura 7.9: Instância Nikkei com $\lambda \in [0{,}10]$.

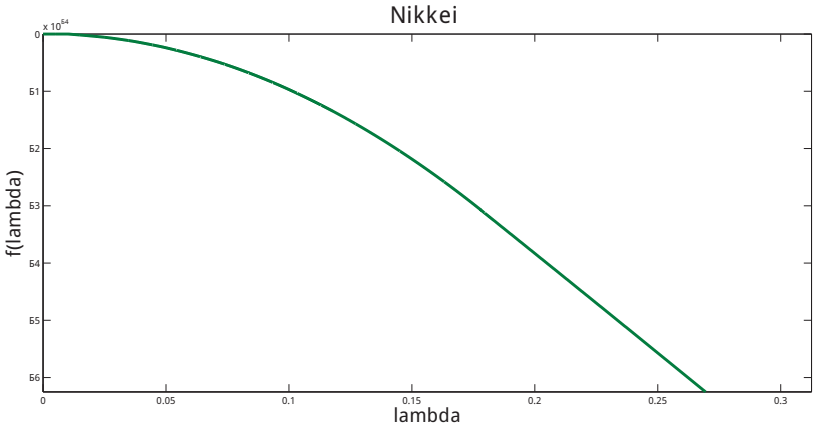


Figura 7.10: Instância Nikkei com $\lambda \in [0; 0{,}3]$.

a FEI é traçada no intervalo [0;0,3], pois nesse caso é possível notar tanto o aspecto quadrático da curva, quanto o segmento, próximo à origem, em que a curva nula é ótima. Entretanto, mesmo nesse caso, as mudanças na inclinação não ficam tão evidentes. Isso se deve ao fato delas ocorrerem em pequenos intervalos de λ, devido à constante troca de carteiras quando estamos próximos de $\lambda = 0$. Essas mudanças só podem ser percebidas de fato quando a curva é visualizada, com zoom máximo, nas proximidades da origem. A Figura 7.11 ilustra esse fato para a instância S&P no intervalo [0;0,08].

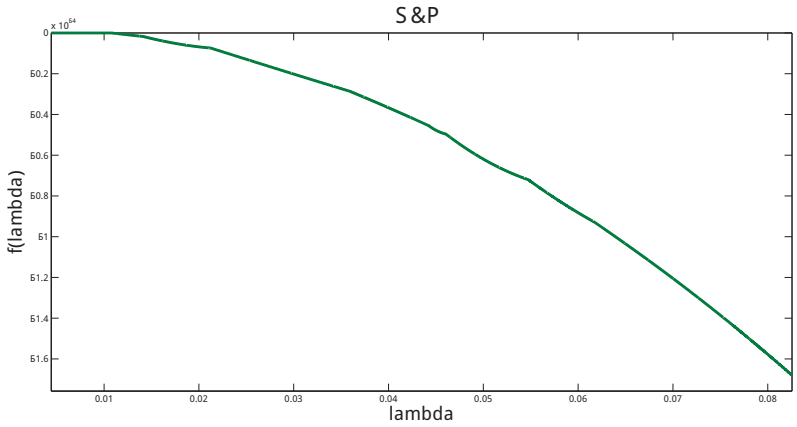


Figura 7.11: Instância S&P com $\lambda \in [0; 0{,}08]$.

7.4 Comparação de fronteiras

Com o intuito de testar a eficiência do nosso algoritmo, decidimos compará-lo com algum outro que aproxima a FEI dos cinco problemas. Nossa ideia original era confrontá-lo com meta-heurísticas eficientes, tais como aquelas propostas por Chang [9] e Dias [13]. Entretanto, não foi possível realizar essa comparação de forma direta, pois a maior parte dos métodos eficientes conhecidos na literatura não só considera que todo o capital é sempre investido, ou seja, que $\sum_{i=1}^{n} x_i = 1$, quanto obtém uma aproximação da FEI em seu formato original, ou seja, no plano em que os eixos representam o risco e o retorno, diferindo um pouco de nossa proposta, que admite que nem todo o montante disponível seja aplicado ($\sum_i^n x_i \leq 1$) e representa a FEI em formato alternativo (a curva $\lambda \times f(\lambda)$).

Por isso, decidimos adaptar o método que desenvolvemos em [36] para discretizar o intervalo $[0,\lambda_{max}]$ em subintervalos de comprimento $\Delta\lambda$ e calcular a solução do Problema (3.1), por meio do Algoritmo BBI, em cada um dos extremos desses intervalos, conforme explicamos nas Seções 2.3

e 3.1. Para comparar os dois métodos quanto à eficiência na aproximação da FEI, desenhamos as curvas obtidas por ambos em um único sistema de eixos.

Como a eficácia da discretização aumenta conforme diminuímos o comprimento dos subintervalos, $\Delta\lambda$, decidimos testar a diferença de aproximação das curvas para uma discretização razoavelmente precisa, $\Delta\lambda = 0{,}1$. Para cada instância, investigamos o que ocorre com as curvas quando λ se encontra próximo de zero, que é o momento em que há trocas constantes de carteira, exigindo assim maior precisão dos nossos métodos. As Figuras 7.12 e 7.13 mostram tanto os resultados obtidos pelo Algoritmo PFV (representado pela curva verde), quanto aqueles obtidos pelo método dos subintervalos, que será chamado de *Algoritmo da Discretização* (representado pela curva azul), para as instâncias Hang Seng e Nikkei no intervalo [0;0,3].

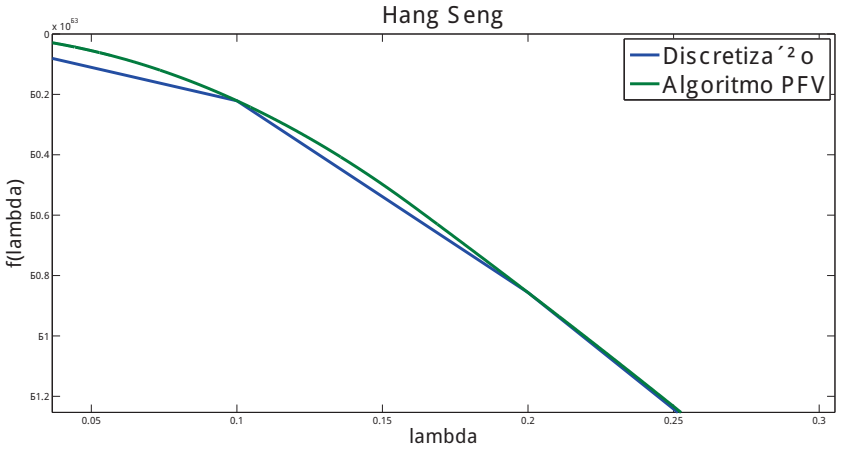


Figura 7.12: Comparação dos métodos para a instância Hang Seng.

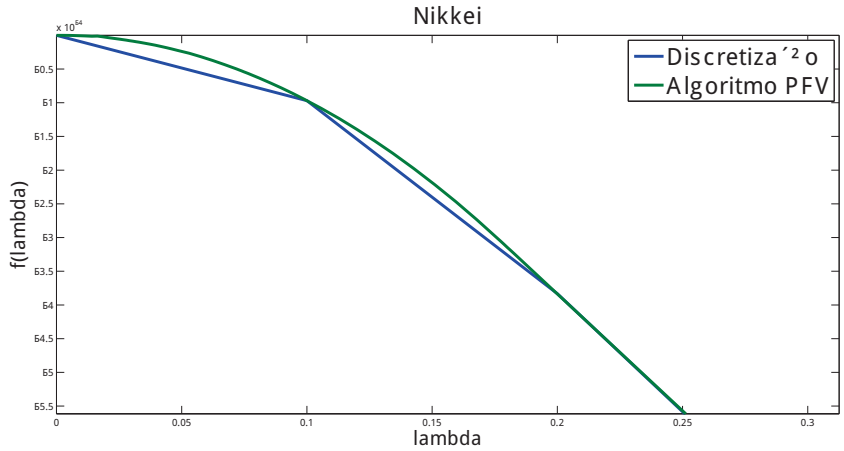


Figura 7.13: Comparação dos métodos para a instância Nikkei.

Podemos notar nas figuras, que as curvas estão relativamente próximas quando λ é maior que 0,2. Porém existe uma clara disparidade entre elas no intervalo [0;0,2]. Isso se deve ao fato do Algoritmo da Discretização não calcular os pontos chave da FEI, que são constantemente trocados quando λ está próximo da origem, fazendo com que a inclinação obtida pelo mesmo se assemelhe a de uma reta, ao passo que a inclinação obtida pelo Algoritmo PFV, que calcula a solução exata, seja de fato a inclinação de uma função quadrática definida por partes. Essa comparação mostra claramente que a aproximação obtida pelo nosso método é mais precisa que aquela encontrada pela técnica da discretização munida de uma precisão razoável, em especial quando λ está próximo de zero, ou seja, esse teste confirma o que era esperado teoricamente.

Em nosso trabalho, fizemos essa mesma comparação com o Algoritmo da Discretização munido de duas discretização mais precisas: 0,01 e 0,0025. Entretanto, observamos que, nos dois casos, a distância entre as curvas diminui drasticamente, conforme diminuímos o valor de $\Delta\lambda$, sendo que a curva obtida pelo Algoritmo da Discretização com $\Delta\lambda = 0{,}0025$ foi praticamente idêntica àquela obtida pelo Algoritmo PFV, mostrando que a aproximação da FEI pelo Algoritmo da Discretização converge para aquela feita pelo nosso algoritmo. Sendo assim, uma vez que as curvas ficaram muito próximas, decidimos não incorporar essa comparação ao nosso trabalho.

7.5 Comparação de tempo computacional

Também comparamos o tempo para aproximar a FEI de cada uma das cinco instâncias, tanto pelo Algoritmo PFV, quanto pelo Algoritmo da Discretização com três discretizações distintas: 0,1, 0,01 e 0,0025. A Tabela 7.2 mostra os resultados obtidos.

Tabela 7.2: Comparação de tempo computacional.

	Hang Seng	Dax	FTSE	S&P	Nikkei
Discretização com $\Delta\lambda$=0,1	0,24 s	0,37 s	0,41 s	110,02 s	2,58s
Discretização com $\Delta\lambda$=0,01	0,76 s	3,59 s	5,66 s	Impossível	26,54 s
Discretização com $\Delta\lambda$=0,0025	2,57 s	12,43 s	15,28 s	Impossível	105,97 s
Algoritmo PFV	1,24 s	8,05 s	8,11 s	326,26 s	6,32 s

Podemos perceber que, para as três primeiras instâncias, o Algoritmo PFV foi mais rápido do que o Algoritmo da Discretização com $\Delta\lambda$=0,0025 e mais lento com as demais discretizações. Tal resultado mostra que, para problemas de pequeno porte que possuem a matriz Hessiana bem

condicionada, o Algoritmo PFV é competitivo com o Algoritmo da Discretização, em termos de tempo computacional, apenas quando exigimos uma grande precisão para a FEI. Já para a quarta instância, que mostrou ser a mais cara de se resolver das cinco, o Algoritmo PFV mostrou seu verdadeiro potencial, ao ser capaz de encontrar a solução em tempo computacional hábil (o tempo limite foi de três horas), ao passo que o Algoritmo da Discretização só foi capaz de resolvê-la para a discretização mais imprecisa. Já para a quinta instância, o Algoritmo PFV só foi mais lento que o Algoritmo da Discretização com a primeira discretização, que é justamente a mais imprecisa. Tal resultado sugere que, conforme a dificuldade e o número de variáveis do problema aumenta, o Algoritmo PFV é mais barato que o Algoritmo da Discretização munido de uma discretização mais precisa, sem contar que a fidelidade da aproximação da FEI obtida por ele é sempre melhor, ou seja, a proposta atual mostrou ser significativamente mais eficiente que a anterior [36].

De qualquer forma, em todos os casos, o tempo do Algoritmo PFV para obter a solução exata dos problemas foi muito pequeno, mostrando que o método, além de obter a solução exata do problema, é possivelmente competitivo com muito outros. Ainda, vale lembrar que o Algoritmo PFV conseguiu determinar, de forma exata, todos os pontos chave dos problemas, que, por si só, são de extrema valia.

7.6 Outros testes de desempenho

Para testar o desempenho geral do nosso algoritmo, realizamos alguns testes extras de tempo computacional. Optamos tanto por aumentar o tamanho da carteira dos problemas clássicos, quanto por realizar testes com instâncias diferentes daquelas já testadas.

7.6.1 Aumentando o tamanho da carteira

Como, na prática, o tamanho máximo da carteira pode variar e investidores mais conservadores podem optar por criar portfólios com mais de 10 ativos, decidimos aplicar o nosso método, ao cálculo da FEI de cada uma das cinco instâncias apresentadas na Seção 7.1, considerando que o número máximo de ativos que podem compor a carteira seja elevado. Tal decisão tem potencial para tornar nosso método mais caro, pois uma diversidade maior de ativos pode gerar um número maior de nós na árvore do Algoritmo BBI. Nesses testes, para cada uma das cinco instâncias, tomamos $K = 25$ e $l_i = 0{,}02$, para $i = 1, ..., n$. A Tabela 7.3 mostra os resultados obtidos.

Tabela 7.3: Teste de tempo computacional para uma carteira maior.

Instância	*K=10*	*K=25*
Hang Seng	1,24 s	0,34 s
Dax	8,05 s	6,38 s
FTSE	8,11 s	7,47 s
S&P	326,26 s	91,01 s
Nikkei	6,32 s	6,82 s

Podemos perceber da tabela que, ao contrário do que era esperado, nas quatro primeiras instâncias o tempo gasto diminuiu, quando aumentamos o tamanho da carteira de 10 para 25, tendo se mantido próximo ao valor obtido anteriormente no caso da última. Esses resultados comprovam que o nosso método é também eficiente para carteiras maiores, podendo assim contemplar as ambições de uma maior variedade de investidores.

7.6.2 Testando outros problemas reais

Com o intuito de testar mais a fundo a eficiência do nosso algoritmo, decidimos explorar os problemas teste apresentados por Cesarone *et al.* [7]. Os problemas selecionados também correspondem a indicadores econômicos baseados em grandes bolsas de valores do mundo, embora diferentes daqueles utilizados por Chang *et al.* [9].

Vale notar que, dentre todas as instâncias propostas, decidimos trabalhar apenas com duas delas, que são justamente aquelas cuja matriz de covariância é simétrica definida positiva e podem ser resolvidas pelo nosso algoritmo. A primeira, denominada EuroStoxx, tem 48 ativos e está relacionada à bolsa europeia. Já a segunda, FTSE, tem 79 ativos cujos dados são diferentes daqueles utilizados nos testes anteriores[1].

Novamente, para cada instância, geramos, por meio do Algoritmo PFV, uma aproximação da FEI. Além disso, restringimos, mais uma vez, a 10 o número de ativos que compõem a carteira ($K = 10$) e utilizamos o limite mínimo de investimento de 5% para cada ativo ($l_i = 0{,}05$, $i = 1, ..., n$). A Tabela 7.4 mostra o tempo gasto pelo Algoritmo PFV na tentativa de resolver cada um dos dois problemas.

[1]Os dados estão disponíveis em `http://w3.uniroma1.it/Tardella/datasets.html`.

Tabela 7.4: Novos testes de tempo computacional.

Instância	Tempo
EuroStoxx	4,10 s
FTSE	12,37 s

Como podemos observar, o Algoritmo PFV mostrou ser eficiente na resolução dos dois problemas, que, por sua vez, possuem um número de variáveis semelhante àqueles propostos por Chang. Tais resultados mostram que o nosso método continua sendo eficiente para problemas cujo número de ativos não é excessivo.

7.7 Trabalhando com restrições de igualdade

Como visto anteriormente, trabalhamos apenas com restrições de desigualdade. Entretanto, uma vez que a análise feita para encontrar a curva ótima não depende da natureza das restrições do problema original, o Algoritmo PFV pode ser adaptado para trabalhar com restrições de igualdade sem maiores dificuldades, de modo a permitir que, no Problema (3.1), todo o capital seja aplicado. Nesse caso, a principal mudança na FEI é o fato de que a aplicação nula deixa de ser ótima para λ próximo de zero, pois ela viola a restrição $\sum_{i=1}^{n} x_1 = 1$. Pelo contrário, a melhor solução passa a ser diversificar o capital, de forma que a solução ótima seja formada pela carteira com os K ativos de menor variância, fazendo com que a solução deixe de ser singular perto da origem, o que torna o problema mais tratável.

Porém, a escolha pelo uso desse tipo de restrição pode acarretar problemas ao longo do Algoritmo PQIM, pois a combinação do *Branch-and-Bound* implícito com o método de *Lemke*, utilizada no Algoritmo PQIM para garantir a integralidade da solução do Problema (3.1) para um valor de λ fixo, só consegue garantir a otimalidade de problemas que envolvem desigualdades. Isso se deve ao fato do método de Lemke só trabalhar com esse tipo de restrição, fazendo com que a conversão de uma restrição de igualdade em duas desigualdades possa gerar um problema de ciclagem, em decorrência da existência de duas colunas linearmente dependentes na matriz do problema, como pode ser observado no trabalho de Villela [36].

Logo, para garantir que o Algoritmo PFV encontre a FEI de problemas que envolvem restrições de igualdade, seria necessário utilizar um outro algoritmo robusto de programação quadrática mista combinado com o método *Branch-and-Bound*. Isso é algo que planejamos fazer em um trabalho futuro.

Capítulo 8

Considerações finais

Analisando os resultados obtidos como um todo, é possível observar que a combinação de nossos três algoritmos principais, quando bem implementada, apresenta um bom desempenho para obter a FEI alternativa de problemas quadráticos baseados no modelo de Markowitz para a otimização de carteiras de investimento. Mais que isso, nosso método se mostrou robusto ao conseguir resolver em tempo hábil, problemas difíceis de médio e grande porte. Ainda, no caso em que desejamos montar uma carteira com um número relativamente pequeno de ativos, o método exato consegue fornecer uma solução ótima em um tempo razoavelmente pequeno.

Vale notar que o Algoritmo PFV, além de aproximar a FEI do Problema (3.1) com boa precisão, é capaz de encontrar os pontos e pesos mais significativos da mesma, que são tanto aqueles que representam a mudança de carteira de investimento, quanto aqueles que representam as mudanças de inclinação da curva, decorrentes da mudança no conjunto ativo das restrições. Essas informações, por si só, podem vir a ser muito úteis ao investidor na hora de analisar as tendências de mercado.

Apesar dos bons resultados obtidos, muitas melhorias podem ser incorporadas ao nosso trabalho. Dentre elas, as que pretendemos investigar em um futuro próximo são

- Otimizar o nosso algoritmo para que ele consiga trabalhar mais rapidamente com valores de λ próximos de zero, fazendo com que ele seja sempre mais rápido do que o *Método 1* com $\Delta\lambda = 0{,}01$.

- Adaptar o Algoritmo PFV para que ele possa resolver problemas que envolvem restrições de igualdade, assim como encontrar a fronteira eficiente de Pareto do problema (3.1) em seu formato original, ou seja, determinar a curva bidimensional em que cada eixo representa o valor dos objetivos calculados nos pontos não dominados. Tal mudança, além de torná-

lo mais próximo da realidade, permitiria uma comparação direta com as meta-heurísticas propostas por Chang [9] e Dias [13].

- Otimizar o desempenho do CHR por meio da estratégia proposta por Guignard e Ahlatcioglu em [26].
- Fazer com que, quando aplicável, o Algoritmo BBI comece sua busca na direção da solução fracionária obtida pelo Algoritmo CHR, que em geral está próxima da solução inteira ótima, eliminando grande parte dos nós pendentes nesse processo.
- Testar diferentes algoritmos de busca na árvore do *Branch and Bound* implícito ligada ao problema com variáveis inteiras.
- Implementar o programa em uma linguagem de alto nível, tal como o C ou C++. Isso não só tornaria mais rápido o algoritmo, como possivelmente tornaria o CHR mais eficiente.
- Aplicar nosso método à solução de problemas reais que não possuam relação com o mercado financeiro, de forma a estendê-lo para um número maior de problemas biobjetivo que envolvem programação quadrática inteira mista.

Referências Bibliográficas

[1] Antczak, T.; *A New Approach to Multiobjective Programming with a Modified Objective Function*. Journal of Global Optimization, 27, p.485-495, 2003.

[2] Audet, C.; Savard, G.; Ghal, W.Z.; *Multiobjective optimization through a series of single-objective formulations*. Siam Journal on Optimization, 19, p.188-210, 2008.

[3] Bazaraa, M.S.; Sherali, H.D.; Shetty, C.M.; *Nonlinear programming: theory and algorithms*. Nova York, John Wiley & Sons Inc, segunda edição, 1979.

[4] Bertsimas, D; Shioda, R.; *An Algorithm for cardinality constrained quadratic optimization*. Computational Optimization and Applications, 43, p.1-22, 2009.

[5] Best, M.J.; *An algorithm for the solution of the parametric quadratic problem*. Applied mathematics and parallel computing, p.57-76, 1996.

[6] Bienstock, D.; *Computacional study of a family of mixed-integer quadratic programming problems*. Mathematical Programming, 74, p.121-140, 1996.

[7] Cesarone, F.; Scozzari, A.; Tardella, F.; *Efficient Algorithm for mean-variance portfolio optimization with hard real-world constraints*. Giornale dell'Istituto Italiano degli Attuari, 72, p.37-56, 2009.

[8] Cesarone, F.; Scozzari, A.; Tardella, F.; *Portfolio selection problems in practice: a comparison between linear and quadratic optimization models*. Artigo encontrado em `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2365855`, acessado no dia 11 de Outubro de 2014.

[9] Chang, T.J.; Maede, N.; Beasley, J.E.; Sharaiha, Y. M.; *Heuristics for cardinality constrained portfolio optimization*. Computer and Operations Research, 27, p.1271-1302, 2000.

[10] Chankong, J.; Haimes, Y.; *Multiobjective Decision Making Theory and Method*. Nova York, Elsevier Science, 1983.

[11] Clarke, F.H; *Optimization and Nonsmooth Analysis*. Nova York, John Wiley & Sons, 1983.

[12] Das, I.; Dennis, J.E.; *Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems*. Siam Journal on Optimization, 8, p.631-657, 1998.

[13] Dias, C.H.; . *Um novo algoritmo genético para a otimização de carteiras de investimento com restrições de cardinalidade*. Dissertação de Mestrado em Matemática Aplicada. Departamento de Matemática Aplicada, Universidade de Campinas, Campinas, Brasil, 2008.

[14] Drummond, L.M.G; Iusem, A,N.; *Projected gradient method for vector optimization problems*. Computational Optimization and Applications, 28, pp. 5–29, 2004.

[15] Drummond, L.M.G; Fukuda, E.H; *On the convergence of the projected gradient method for vector optimization*. Optimization: A Journal of Mathematical Programming and Operations Research, 60, p.1009-1021, 2011.

[16] Drummond, L.M.G; Sveiter, B.F.; *A steepest descent method for vector optimization*. Journal of Computational and Applied Mathematics 175, p.395–414, 2005.

[17] Eichfelder, G.; *An adaptative scalarization mathod in multiobjective optimization*. Siam Journal on Optimization, 19, p.1644-1718, 2009.

[18] Ehrgott, M.; Ruzika, S.; *An improved ϵ-constraint method for multiobjective programming*. Journal of Optimization Theory and Applications, 138, p.375-396, 2008.

[19] Ehrgott, M.; Waters, C.; *Multiobjective programming and multiattribute utility functions in portfolio optimization*. Information Systems and Operational Research, 47, p.1-42, 2009.

[20] Fernández, J.; Tóth, B.; *Obtaining the efficient set of nonlinear biobjective optimization problems via branch-and-bound methods*. Computational Optimization and Applications, 42, p.393-419, 2009.

[21] Fliege, J; Drummond, M.G.; Sveiter, B.F.; *Newton's Method for Multiobjective Optimization*. Siam Journal on Optimization, 20, p.602-626, 2009.

[22] Fliege, J.; Heseler, A.; *Constructing Approximations to the Efficient Set of Convex Quadratic*. University of South Hampton, 2004.

[23] Fliege, J.; *An Efficient Interior-Point Method for Convex Multicriteria Optimization Problems*. Mathematics of Operations Research, 31, 2006.

[24] Fliege, J.; Sveiter, B.F.; *Steepest Descent Methods for Multicriteria Optimization*. Mathematical Methods of Operations Research. 51, p.479-494, 2000.

[25] Frank, M.; Wolfe, P.; *An Algorithm for quadratic Programming*. Naval Research Quarterly, 3, p.95-109, 1956.

[26] Guignard, M.; Ahlatçioglu, A; *The Convex Hull Relaxation for Nonlinear Integer Programs with Convex Objective and Linear Constraints*. European Workshop in Mixed-Integer Nonlinear Programming, França, 2010.

[27] Kim, I.Y.; Weck, O.L.; *Adapatative weighting-sum method for bi-objective optimization Pareto front generation*. Structural and Multidisciplinary Optimization, 29, p.149-158, 2005.

[28] Leyffer, S.; *A note on multiobjective optimization and complementarity constraints*. Journal on Computing, 21, p.257-267, 2009.

[29] Luenberger, D. G.; *Investment Science*. Nova York, Oxford University press, 1998.

[30] Markowitz, H.; *Portfolio Selection*. Journal of Finance, 7, p.77-91, 1952.

[31] Mirttinen, K.; Makela, M.M.; *On cone characterizations of weak, proper and Pareto optimality in multiobjective optimization*. Mathematical Methods of Operation Research, 53, p.233-245, 2001.

[32] Mitra, G.; Kyriakis, T.; Lucas, C.; *A review of portfolio planning: Models and systems*. Primeiro capítulo de Advances in Portfolio Construction and Implementation, Oxford, Butterworth and Heinmann, p. 1-39, 2003.

[33] Pareto, V.; *Manuale di Economia Politica*. Italia, Societa Editrice Libraria, 1906.

[34] Plas, C.P.; Tervonen, T.; Dekker, R.; *Evaluation of scalarization methods and NSGA-II/SPEA2 genetic algorithms for multi-objective optimization of green supply chain design*. Report Econometric Institute, 2012.

[35] Stein, M.; *Efficient Implementation of an active set algorithm for large-scale portfolio selection*. Computers and Operations Research, 35, p.3945-3961, 2008.

[36] Villela, P.F.; *Um algoritmo exato para a otimização de carteiras de investimento com restrições de cardinalidade*. Dissertação de Mestrado em Matemática Aplicada. Departamento de Matemática Aplicada, Universidade de Campinas, Campinas, Brasil, 2008.

[37] Zadeh, L.; *Optimality and Non-Scalar-Valued Performance Criteria*. IEEE Transactions on Automatic Control, 8, p.59-60, 1963

Apêndice A

Métodos e conceitos

A.1 O método *Branch-and-Bound* para problemas 0-1

O *Branch-and-Bound* é um algoritmo baseado no processo de divisão e conquista, que consiste em dividir um problema difícil (em geral, NP-completo) em vários problemas menores, de fácil resolução, para posteriormente juntar as informações obtidas e assim resolver o problema original. Dessa forma, é feita uma enumeração sistemática das possíveis soluções, procurando sempre eliminar, ao longo do caminho, grupos de soluções menos proveitosas. Em geral, esse algoritmo é muito utilizado para achar a solução ótima de problemas de otimização que envolvem variáveis inteiras.

Neste trabalho, estamos interessados em resolver problemas de programação quadrática inteira mista, em que algumas variáveis são binárias, ou seja, só podem assumir valores 0 ou 1. Para resolver esse tipo de problema usando o *Branch-and-Bound*, primeiramente resolvemos a versão relaxada do problema, o que no nosso caso corresponderia a resolver o problema no qual todas as variáveis binárias são consideradas contínuas no intervalo [0,1]. Feito isso, verificamos se, na solução obtida, todas as variáveis binárias valem 0 ou 1. Se isso ocorre, já resolvemos o problema original. Caso contrário, como ainda não encontramos uma solução viável para o problema original, guardamos uma solução vazia como a melhor solução inteira. Neste caso, definindo ζ^* como o valor da função objetivo desta melhor solução, tomamos $\zeta^* \leftarrow \infty$.

Em seguida, escolhemos uma das variáveis binárias que por ora possui valor fracionário e a forçamos a assumir explicitamente cada um dos dois valores admissíveis (0 e 1). Geramos, assim, dois subproblemas que precisam ser resolvidos. A essa subdivisão do problema damos o nome de ramificação. Nesse contexto, existem 2 tipos diferentes de ramificação: a para baixo e a para cima. Na ramificação para baixo, a variável escolhida assume o valor 0, ao passo que na ramificação para cima ela assume o valor 1.

É fácil observar que cada ramificação dá origem a outras ramificações, até que todas as variáveis inteiras da solução obtida possuam valor 0 ou 1. Assim, a estrutura do problema sugere a criação de uma árvore binária para poder representar todas as possíveis ramificações a serem investigadas.

Nessa árvore, cada subproblema gerado por uma ramificação é chamado de *nó*. O primeiro problema relaxado a ser resolvido é conhecido como *nó raiz*. Um nó cuja solução tenha as variáveis binárias valendo 0 ou 1 é chamado de *folha*, ao passo que um nó em que algumas das variáveis binárias estejam em (0,1) é chamado de *ramo*. Quando um nó i gera um nó j via ramificação, dizemos que o nó i é o *pai* de j, de modo que este último é um nó *filho* de i. Um nó que ainda não foi resolvido é chamado de *nó pendente*. O número máximo de elementos da árvore é igual a $2^{N+1}-1$, onde N é o número de variáveis binárias do problema.

Depois de resolver o problema associado a uma ramificação, verificamos se a solução é factível, no sentido das variáveis binárias do problema assumirem apenas valores 0-1. Se isso ocorre, comparamos ζ, o valor da função objetivo, com ζ^*, o valor de ζ associado à melhor solução inteira encontrada até o momento. Se $\zeta \leq \zeta^*$, atualizamos ζ^* e deixamos de resolver todas as ramificações pendentes cuja solução seja pior que aquela que acabamos de encontrar. Por outro lado, se a solução ainda incluir variáveis binárias com valores não inteiros, escolhemos uma nova variável para ramificar, gerando dois novos problemas.

Salvo quando aparece uma folha, a cada iteração do método, temos que armazenar 2 novos nós e decidir qual nó resolver dentre os possíveis pendentes. O número de nós pendentes só diminui quando achamos uma folha, pois, neste caso, além de não ramificarmos, eliminamos os nós pendentes com valor de ζ maior que o da solução recém encontrada. O algoritmo acaba quando não existem mais nós pendentes para investigarmos.

A.2 O problema de complementaridade linear

Dados $q \in \mathbb{R}^n$ $M \in \mathbb{R}^{n_x n}$, um problema de complementaridade linear (PCL) consiste em encontrar $\psi \in \mathbb{R}^n$ e $\omega \in \mathbb{R}^n$ tais que:

$$\begin{cases} \omega = M\psi + q \\ \psi \geq 0, \omega \geq 0 \\ \psi^T \omega = 0. \end{cases} \tag{A.1}$$

Esse problema é conhecido como *PCL(q,M)*. Um vetor ψ é dito factível se satisfaz as duas primeiras restrições de (A.1), e é dito complementar se satisfaz a última equação. Todo vetor ψ que é factível e complementar é chamado de ponto de equilíbrio do *PCL(q,M)*. Note que, devido

às restrições de não negatividade, ψ e ω são complementares se e somente se

$$\psi_i \omega_i = 0, \forall i = 1,2,...n.$$

Chamamos o par (ψ_i,ω_i) de *par complementar*.

O método de Lemke, que em nosso trabalho foi utilizado para otimizar os problemas relaxados associados aos nós da árvore gerada pelo algoritmo *Branch-and-Bound*, foi criado originalmente para resolver esse tipo de problema.