UNIVERSIDADE D
COIMBRA

Luís Henrique Rodrigues Duarte

Aplicação de métodos biobjetivo a otimização linear fracionária

Dissertação no âmbito do Mestrado em Matemática, área de especialização Estatística, Otimização e Matemática Financeira orientada pela Professora Doutora Marta Pascoal e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia.

Setembro de 2019

Aplicações de métodos biobjetivo a otimização linear fracionária

Luís Henrique Rodrigues Duarte

UNIVERSIDADE D
COIMBRA

Mestrado em Matemática

Master in Mathematics

Dissertação de Mestrado | MSc Dissertation

2019

Agradecimentos

Esta Dissertação foi redigida naquele que, sem dúvida, foi o ano mais desafiante da minha vida académica. Assim sendo, não posso deixar de fazer alguns agradecimentos. Em primeiro lugar, à minha orientadora, a Professora Doutora Marta Pascoal, por toda a dedicação a este projeto, a ajuda que nunca me negou e a paciência que teve para comigo ao longo deste ano. Ao meus colegas de curso, por todo o apoio e confiança que sempre depositaram em mim e, como não podia deixar de ser, ao Núcleo de Estudantes de Matemática da Associação Académica de Coimbra por me fazer ser um melhor profissional, aumentando o meu sentido de responsabilidade. Aos meus pais e à minha irmã por estarem sempre ao meu lado e apoiarem todas as minhas decisões ao longo do meu percurso. Sendo este projeto realizado em 2019, não me poderia esquecer da Direção Geral da Associação Académica de Coimbra. A todos os meus colegas de equipa e principalmente aos que são mais do que isso, um enorme obrigado!

Este trabalho foi desenvolvido no âmbito do projeto "MobiWise: From mobile sensing to mobility advising" (P2020 SAICTPAC/0011/2015), co-financiado pelo COMPETE 2020, Portugal 2020 - Programa Operacional para a Competitividade e Internacionalização, o Fundo Europeu de Desenvolvimento Regional, e a Fundação para a Ciência e Tecnologia.

Cofinanciado por:

COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional FCT Fundação para a Ciência e a Tecnologia

Resumo

O tema desta dissertação é a otimização linear fracionária. Em primeira instância, são apresentados e testados diversos métodos (iterativos e não iterativos) conhecidos na literatura para a resolução de problemas lineares fracionários. Em segunda instância, propõe-se a resolução do mesmo problema através de dois algoritmos baseados em métodos para otimização biobjetivo, nomeadamente em métodos de somas ponderadas para o cálculo de soluções eficientes suportadas de um problema biobjetivo associado a um programa linear fracionário. O comportamento empírico dos dois conjuntos de algoritmos é estudado e comparado para programas lineares fracionários e problemas da mochila, nas versões binária e relaxada, com várias dimensões. Em geral, os métodos paramétricos revelaram-se mais eficientes do que os métodos biobjetivo para os problemas de teste. A segunda versão biobjetivo apresentada, método da soma ponderada modificado, mostrou-se eficiente para os problemas considerados. Em geral, os métodos paramétricos mostraram-se mais rápidos do que esta versão, contudo, o método da soma ponderada modificado foi competitivo na maioria dos casos e, portanto, pode ser considerado uma alternativa interessante aos métodos conhecidos na literatura.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os problemas de programação fracionária são problemas de otimização tendo um quociente de duas funções como função objetivo. Algumas extensões destes problemas consideram a otimização de somas de quocientes daquele tipo, ou a otimização simultânea de vários destes quocientes. Quando a região admissível é definida por um conjunto de restrições lineares e o quociente envolve duas funções afins, o problema diz-se um programa linear fracionário. É este o objeto de estudo do presente trabalho.

Os programas lineares fracionários têm diferentes tipos de aplicações. Destas destacam-se aplicações diretas, de índole económica, como por exemplo a maximização, ou minimização, de um custo por unidade de tempo [1, 2, 10, 13–15, 21, 31, 36], a maximização do retorno sobre o investimento, certos problemas de alocação de recursos [26, 27], ou a maximização das proporções lucro/capital ou lucro/receita. Utiliza-se igualmente programação fracionária na área de apoio à decisão, nomeadamente em análise de envolvimento de dados[1], para avaliar o desempenho de um conjunto de unidades de tomada de decisão em termos relativos [4, 6, 9], e em otimização difusa, um dos paradigmas utilizado para lidar com problemas para os quais existe incerteza em alguns dos parâmetros [16, 24].

Por vezes, programas lineares fracionários também surgem de forma indireta, isto é, relacionados com a resolução de outros problemas de otimização. Por exemplo, a resolução de problemas lineares de grande dimensão pode conduzir a subprogramas fracionários, decorrentes da aplicação da regra do quociente mínimo do método simplex [22]. Do mesmo modo, os programas lineares fracionários são utilizados em programação estocástica, quando, considerando um modelo em que se procura maximizar a probabilidade de uma função de uma variável aleatória atingir um determinado valor, conforme introduzido por Charnes e Cooper, e por Bereanu [3, 5, 39]. Alguns exemplos destes e de outros problemas de otimização linear fracionária podem ser encontrados em trabalhos como [24, 29, 32–34].

Em [32] Schaible e Ibaraki apresentam uma revisão bibliográfica sobre programação fracionária, enquanto que em [30] Radzik se foca, particularmente, em programação fracionária para problemas de otimização combinatória. Recomenda-se ainda o trabalho [37], de Stancu-Minasian, o mais recente de uma série de publicações que compila e classifica referências atuais sobre programação fracionária. Além destas publicações, de índole mais geral, outros autores têm abordado problemas de otimização

[1] Do inglês *Data Envelopment Analysis* (ou DEA).

combinatória de um ponto de vista fracionário. Aos trabalhos de otimização linear fracionária mencionados anteriormente e que tratam aplicações, acrescentam-se outros sobre o problema da afetação [20, 23, 35], o problema do caminho mais curto por unidade de tempo [38], ou o problema da mochila [19, 29], entre outros.

Apesar da não linearidade dos programas lineares fracionários, Charnes e Cooper [5] propuseram uma mudança de variável que transforma um problema deste tipo noutro que é linear. Este processo permite resolver muitos problemas lineares fracionários, contudo tem algumas limitações e desvantagens. Por exemplo, o facto de não se aplicar quando as variáveis de decisão são inteiras e de conduzir a métodos de resolução pouco eficientes, para problemas cujas soluções podem ser calculadas facilmente através de métodos especializados. Os algoritmos paramétricos, como os introduzidos por Isbell e Marlow [18], Dinkelback [11] ou Ibaraki [17], têm sido alternativas muitos populares à mudança de variável de Charnes e Cooper, e calculam de forma iterativa o zero de uma função não linear que depende de um parâmetro variável. Em termos práticos estes algoritmos resolvem uma sequência de problemas da mesma classe do dado, mas com uma função objetivo linear, dependente dum parâmetro que é atualizado com base nas soluções já determinadas. Em geral estes métodos são conhecidos pela simplicidade de implementação, a versatilidade e o facto de convergirem para a solução ótima em poucas iterações.

A minimização (maximização) de um quociente pode ser formulada de forma intuitiva como um problema biojetivo em que se minimiza (maximiza) o numerador e maximiza (minimiza) o denominador. O principal propósito deste trabalho é estudar a aplicabilidade de métodos para o cálculo de soluções eficientes do problema linear biobjetivo associado à resolução de programas lineares fracionários. Para este efeito introduzem-se dois algoritmos. O primeiro resulta da adaptação direta de um método clássico para encontrar soluções eficientes suportadas de problemas biobjetivo, baseado em somas ponderadas cujos parâmetros são ajustados pelos pares de soluções adjacentes já calculadas. O segundo inclui informação sobre a evolução dos parâmetros utilizados para melhorar a versão inicial do algoritmo, diminuindo o número de soluções que é necessário determinar e, consequentemente, o número de iterações realizadas e o tempo de execução do método. Os resultados teóricos apresentados são complementados por uma avaliação computacional dos dois algoritmos para problemas aleatórios com diferentes dimensões.

O restante texto está estruturado do seguinte modo. O Capítulo 2 é dedicado à formulação de programas lineares fracionários e à revisão bibliográfica de resultados e algoritmos conhecidos para o resolver. A revisão compreende métodos baseados em duas abordagens distintas: uma paramétrica, ou iterativa, e outra não iterativa. Inclui-se, ainda, experiência computacional para um conjunto de problemas gerados de forma aleatória. No capítulo seguinte são introduzidos dois métodos biobjetivo para programas lineares fracionários. Para além disto, apresentam-se resultados de experiência computacional envolvendo implementações desses mesmos métodos, quando aplicadas a um conjunto de problemas de teste com várias dimensões, em condições semelhantes às do Capítulo 2. O Capítulo 4 é dedicado a um problema combinatório clássico, o problema da mochila. Após uma breve discussão sobre o problema da mochila e sobre algumas das suas versões, os algoritmos paramétricos e os algoritmos biobjetivo abordados nos Capítulos 2 e 3 são comparados quando aplicados a este problema. Para o efeito consideram-se uma versão do problema da mochila fracionário com variáveis binárias e uma versão relaxada desta. O Capítulo 5 contém alguns comentários finais ao trabalho.

Capítulo 2

Programas lineares fracionários

No presente capítulo abordam-se problemas de otimização em que se pretende minimizar, ou maximizar, uma função objetivo que é o quociente de duas funções. Descrevem-se duas classes de métodos para resolver estes problemas: os paramétricos, que podem ser aplicados a qualquer quociente, e uma mudança de variável, que permite lidar com programas fracionários ditos lineares, em que as funções no numerador e no denominador da função objetivo são afins. O capítulo é concluído com a comparação computacional dos métodos descritos para um conjunto de programas lineares fracionários gerados aleatoriamente.

2.1 Definição e propriedades

Seja E^n um espaço Euclidiano com dimensão n e S um subconjunto compacto de E^n. Além disso, suponha-se que $N(x)$ e $D(x)$ são funções reais e contínuas em S tal que $D(x) > 0$, para todo o $x \in S$, e considere-se o problema

$$\min \left\{ \frac{N(x)}{D(x)} \mid x \in S \right\}. \tag{2.1}$$

Quando as funções N e D são quaisquer, este problema é designado de programa fracionário. No caso de N e D serem funções afins, diz-se um programa linear fracionário, o qual se tratará adiante.

Uma das abordagens conhecidas para resolver este problema consiste em transformá-lo numa sequência de outros problemas, dependentes de um parâmetro q, em que o quociente na função objetivo é eliminado [11]. Considera-se então

$$\min \{N(x) - qD(x) \mid x \in S\}, \text{ com } q \in E^1. \tag{2.2}$$

Se a região admissível S é não vazia, os problemas (2.1) e (2.2) têm solução, devido ao facto de se terem excluído os pontos que satisfazem $D(x) = 0$. Adicionalmente, verificam-se os resultados que se seguem.

Lema 1. *A função com expressão* $F(q) = \min \{N(x) - qD(x) \mid x \in S\}$ *é côncava em* E^1.

Demonstração. Seja x_t um minimizante de $F(tq' + (1-t)q'')$ com $q' \neq q''$ e $0 \leq t \leq 1$.

$$\begin{aligned} F(tq' + (1-t)q'') &= N(x_t) - (tq' + (1-t)q'')D(x_t) \\ &= t\left[N(x_t) - q'D(x_t)\right] + (1-t)\left[N(x_t) - q''D(x_t)\right] \\ &\geq t \min\left\{N(x) - q'D(x) \mid x \in S\right\} + (1-t)\min\left\{N(x) - q''D(x) \mid x \in S\right\} \\ &= tF(q') + (1-t)F(q''). \end{aligned}$$

□

Lema 2. *A função $F(q)$ é contínua para $q \in E^1$.*

Lema 3. *A função $F(q) = \min\{N(x) - qD(x) \mid x \in S\}$ é estritamente monótona decrescente.*

Demonstração. Pretende-se mostrar que quaisquer que sejam $q', q'' \in E^1$, se $q' < q''$, então $F(q'') < F(q')$. Seja x'' um minimizante de $F(q'')$, então,

$$\begin{aligned} F(q'') &= \min\left\{N(x) - q''D(x) \mid x \in S\right\} \\ &= N(x'') - q''D(x'') \\ &\leq \min\left\{N(x) - q'D(x) \mid x \in S\right\} \\ &= F(q'). \end{aligned}$$

□

Lema 4. *A equação $F(q) = 0$ tem uma única solução.*

Demonstração. Este resultado é consequência dos lemas anteriores, observando ainda que $\lim_{q \to -\infty} F(q) = +\infty$ e $\lim_{q \to +\infty} F(q) = -\infty$. □

Lema 5. *Sejam $x^+ \in S$ e $q^+ = \frac{N(x^+)}{D(x^+)}$, então $F(q^+) \leq 0$.*

Demonstração. Nas condições enunciadas,

$$\begin{aligned} F(q^+) &= \min\left\{N(x) - q^+D(x) \mid x \in S\right\} \\ &\leq N(x^+) - q^+D(x^+) \\ &= 0. \end{aligned}$$

□

Para algum parâmetro fixo $q = q^*$, o mínimo de $N(x) - q^*D(x)$ em S é atingido no ponto $x = x^*$, obtendo-se assim o valor objetivo $F(q^*)$. É então possível mostrar o seguinte teorema.

Teorema 1. *Tem-se $q_0 = \min\left\{\frac{N(x)}{D(x)} \mid x \in S\right\}$ se e só se $F(q_0) = \min\{N(x) - q_0D(x) \mid x \in S\} = 0$.*

Demonstração. Para mostrar a implicação direta, começa por se considerar x_0 uma solução ótima do problema (2.1) e $q_0 = \dfrac{N(x_0)}{D(x_0)}$. Então,

$$q_0 \leq \frac{N(x)}{D(x)}$$

para todo o $x \in S$, o que resulta em

$$N(x) - q_0 D(x) \geq 0, \tag{2.3}$$

para todo o $x \in S$ e

$$N(x_0) - q_0 D(x_0) = 0. \tag{2.4}$$

De acordo com (2.3), tem-se $F(q_0) = \min\{N(x) - q_0 D(x) \mid x \in S\} = 0$, logo o mínimo do problema (2.2) é atingido em x_0.

Seja agora x_0 uma solução do problema (2.2) tal que $N(x_0) - q_0 D(x_0) = 0$. A definição de (2.2) implica

$$N(x) - q_0 D(x) \geq N(x_0) - q_0 D(x_0) = 0$$

para todo o $x \in S$. Então, verificam-se novamente (2.3) e (2.4), tal como no primeiro caso. Com base em (2.3) tem-se $q_0 \leq \frac{N(x)}{D(x)}$, para todo o $x \in S$, tal que q_0 é um mínimo do problema (2.1). Por (2.4) tem-se $q_0 = \frac{N(x_0)}{D(x_0)}$, tal que x_0 é uma solução de (2.1), donde segue o resultado enunciado. □

Note-se que x_0 pode não ser a única solução do problema e que este teorema se aplica tanto nos casos de minimização como nos de maximização.

2.2 Abordagem paramétrica

Em seguida revêm-se alguns métodos paramétricos, processos muito comuns na resolução de problemas fracionários como o problema (2.1) e que resultam de métodos iterativos para a resolução da equação $F(q) = 0$, relativamente ao parâmetro q. Suponha-se agora que $N(x)$ é uma função côncava e que $D(x)$ é uma função convexa, para qualquer $x \in S$. Considere-se ainda que S é um conjunto convexo e denote-se por x_0 uma solução ótima do problema (2.1).

Método de Dinkelbach Escolha-se q_k suficientemente próximo de q^*. Sabendo que $F(q)$ é uma função contínua, obtém-se uma segunda formulação. Isto é, x_k e $q_k = \frac{N(x_k)}{D(x_k)}$ tal que $F(q_k) - F(q_0) = F(q_k) < \delta$, para algum $\delta > 0$. Em seguida, considera-se $F(0) = \min\{N(x) \mid x \in S\} \geq 0$ e o algoritmo inicia-se com $q = 0$ [11]. Na prática, o cálculo é interrompido quando q_k está suficientemente próximo do valor ótimo, isto é, se $|F(q_k)| < \delta$, para alguma tolerância $\delta \in \mathbb{R}^+$ previamente definida. Este método é resumido no Algoritmo 1.

De forma a mostrar a convergência do método, começa-se por verificar que $q_{k+1} > q_k$, para qualquer k, tal que $F(q_k) \geq \delta$. Sabe-se que $F(q_k) > 0$ e, por definição, tem-se $N(x_k) = q_{k+1} D(x_k)$. Consequentemente,

$$\begin{aligned} F(q_k) &= N(x_k) - q_k D(x_k) \\ &= q_{k+1} D(x_k) - q_k D(x_k). \end{aligned}$$

Então, como $D(x_k) > 0$, tem-se $q_{k+1} > q_k$.

O segundo passo consiste em mostrar que

$$\lim_{k \to \infty} q_k = q_0.$$

Algoritmo 1: Método de Dinkelbach

$q_2 \leftarrow 0$
$k \leftarrow 2$
$F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1}) \mid x \in S\}$
while $|F(q_k)| \geq \delta$ **do**
    $q_{k+1} \leftarrow \frac{N(x_k)}{D(x_k)}$
    $k \leftarrow k+1$
    $F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1}) \mid x \in S\}$
return x_k

Supondo que isto não se verifica, tem-se $\lim_{k\to\infty} q_k = q^* < q_0$. Por construção, obtém-se uma sucessão $\{x_k^*\}$ com q_k^*, tal que $\lim_{k\to\infty} F(q_k^*) = F(q^*) = 0$. Sendo $F(q)$ uma função estritamente monótona decrescente, obtém-se $0 = F(q^*) = F(q_0)$, o que é uma contradição. Por fim, como $F(q)$ é contínua, tem-se

$$\lim_{k\to\infty} F(q_k) = F(q_0) \text{ e } \lim_{k\to\infty} q_k = q_0,$$

para $q \in E^1$.

Considerando novamente os problemas (2.1) e (2.2). Se $N(x)$ e $D(x)$ são funções afim, então $N(x) - qD(x)$ também o é, ao contrário de $\dfrac{N(x)}{D(x)}$. Analogamente, se $N(x)$ é côncava, $D(x)$ é convexa e $q \geq 0$, então $N(x) - qD(x)$ é côncava, mas $\dfrac{N(x)}{D(x)}$ não o é.

Seja x^* uma solução ótima de (2.1) e $q^* = \dfrac{N(x^*)}{D(x^*)}$. Então,

1. $F(q) > 0$ se e só se $q < q^*$;
2. $F(q) = 0$ se e só se $q = q^*$;
3. $F(q) < 0$ se e só se $q > q^*$;

e uma solução ótima de (2.2), sendo $q \equiv q^*$, é também uma solução ótima de (2.1). Conclui-se então que resolver (2.1) é equivalente a encontrar o parâmetro $q = q^*$ tal que $F(q) = 0$, sabendo-se que a função F satisfaz as propriedades enunciadas nos Lemas 1 a 5.

Método de Newton Generalizando o método anterior, considere-se q_2 como sendo o valor inicial de q e a reta definida por

$$N(x) - q_2 D(x) = 0. \tag{2.5}$$

Esta reta é tangente a $F(q)$ no ponto $q = q_2$, sendo x^* a solução ótima de (2.2). Para além disto, (2.5) interseta a reta $F(q) = 0$ quando $q = \dfrac{N(x^*)}{D(x^*)}$. Pode mostrar-se que o método de Newton gera uma sucessão $\{q_k\}$ convergindo para q^* [11].

Método da procura binária Considere-se um intervalo $[q_1, q_2]$ que contém q^*. O método da procura binária, semelhante à aplicação do método da bisseção para determinar q^*, é resumido no

Algoritmo 2. Como intervalo inicial devem escolher-se q_1 e q_2 tais que $F(q_1)F(q_2) < 0$ Geralmente, considera-se $q_1 = 0$ e q_2 um valor suficientemente grande, de forma a que o intervalo $[q_1, q_2]$ inclua a solução procurada [17].

Algoritmo 2: Método da procura binária

1 $k \leftarrow 2$
2 $x_1 \leftarrow 0$
3 $q_k \leftarrow \frac{1}{2}(q_1 + q_2)$
4 $F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1})\}$
5 **while** $|F(q_k)| > \delta$ **do**
6 **if** $F(q_k) > 0$ **then** $q_1 \leftarrow q_k$
7 **else** $q_2 \leftarrow q_k$
8 $k \leftarrow k + 1$
9 $q_k \leftarrow \frac{1}{2}(q_1 + q_2)$
10 $F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1})\}$
11 **return** x_k

Ao fim de k iterações deste algoritmo, a amplitude do intervalo $[q_1, q_2]$ não excede $\dfrac{(q_2 - q_1)}{2^k}$, logo este valor é um majorante do erro cometido pelo método de procura binária.

Método da procura binária modificado Este processo resulta da combinação dos métodos de Newton e de procura binária, descritos anteriormente. Sejam q_1 e q_2 tais que $F(q_1) > 0$ e $F(q_2) < 0$, e x^* uma solução ótima de (2.2), em função de q_1. A partir da concavidade de $F(q)$ conclui-se que

$$q = \frac{N(x^*)}{D(x^*)} \quad \text{tal que} \quad q \leq q^*.$$

Seja $v(q_1, q_2)$ o valor de q quando a reta que liga os pontos $(q_1, F(q_1))$ e $(q_2, F(q_2))$ interseta a reta $F(q) = 0$. Tem-se então $q = v(q_1, q_2)$, com $q \geq q^*$. Assim sendo, a primeira modificação em relação ao método da procura binária é o intervalo que contém q^*, definido por

$$\left[\frac{N(x^*)}{D(x^*)}, v(q_1, q_2)\right].$$

A convergência superlinear deste método é alcançada recorrendo, sempre que necessário, ao método de Newton. Tal é possível definindo

$$q = \frac{d(k) - 1}{d(k)} q_1 + \frac{1}{d(k)} q_2, \tag{2.6}$$

em vez da atualização utilizada pelo método de procura binária tradicional, sob a condição de a função $d(k)$ ser crescente, ou seja, $d(k) \geq 2$ e $\lim_{k \to \infty} d(k) = \infty$. Exemplo destas funções são $d(k) = k + 1$ ou $d(k) = 2^k$ [17].

Utilizar o resultado (2.6) na linha 18 do Algoritmo 3 permite estabelecer iterações mais próximas do valor ótimo. No caso das linhas 10 a 13, o processo remete para o método de Newton.

Algoritmo 3: Método da procura binária modificado

$k \leftarrow 2$
$n \leftarrow 1$
$x_1 \leftarrow 0$
$q_k \leftarrow q_1$
$F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1})\}$
$q_k^a \leftarrow \frac{N(x_k)}{D(x_k)}$; $q_k^b \leftarrow v(q_1, q_2)$
$q_k \leftarrow \frac{1}{2}(q_k^a + q_k^b)$
$F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1})\}$
while $|F(q_k)| > \delta$ **do**
    if $F(q_k) > 0$ **then**
        $q_1 \leftarrow q_k$
        $q_k^a \leftarrow \frac{N(x_k)}{D(x_k)}$
        $q_k^b \leftarrow v(q_1, q_2)$
    else
        $q_2 \leftarrow q_k$
        $n \leftarrow n + 1$
        $q_k^b \leftarrow v(q_1, q_2)$
        $q_k \leftarrow \dfrac{d(n) - 1}{d(n)} q_k^a + \dfrac{1}{d(n)} q_k^b$
    $k \leftarrow k + 1$
    $q_k \leftarrow \frac{1}{2}(q_k^a + q_k^b)$
    $F(q_k) \leftarrow \min\{N(x_{k-1}) - q_k D(x_{k-1})\}$
return x_k

Para verificar a rapidez de convergência deste método, considere-se a sucessão $\{q_k\}$ definida nas linhas 7 e 20 do Algoritmo 3. Defina-se a subsucessão que satisfaz $F(q_k) > 0$ como $\{q_k^a\}$, e a subsucessão que satisfaz $F(q_{k-1}) < 0$ e $F(q_k) < 0$ como $\{q_k^b\}$. A subsucessão contendo os restantes valores define-se por $\{q_k^c\}$.

Lema 6. *Se a sucessão $\{q_k^a\}$ é infinita, então*

$$\lim_{k\to\infty}(q^* - q_{k+1}^a)(q^* - q_k^a) = 0,$$

isto é, $\{q_k^a\}$ converge superlinearmente para q^*.

Demonstração. Note-se que, para todo o k tal que $q_k^a < q^*$, se tem $F(q_k^a) > 0$. Sabe-se também que

$$\frac{q^* - \frac{N\left(x^{(k)}\right)}{D\left(x^{(k)}\right)}}{q^* - q_k^a} \leq \frac{D(x^*)}{D\left(x^{(k)}\right)},$$

onde x^* e $x^{(k)}$ são soluções ótimas de (2.1) e (2.2), respetivamente. Uma vez que

$$\frac{N\left(x^{(k)}\right)}{D\left(x^{(k)}\right)} < q_{k+1}^a < q^*,$$

pelo Algoritmo 3, tem-se

$$\frac{q^* - q_{k+1}^a}{q^* - q_k^a} \leq 1 - \frac{D(\tilde{x})}{D\left(x^{(k)}\right)}.$$

Sabe-se que $D\left(x^{(k)}\right)$ converge para $D(x^*)$ quando $q_k^a \to q^*$. Sendo $-D\left(x^{(k)}\right)$ um subgradiente de $F(q)$ em q_k^a e $F(q)$ uma função contínua, conclui-se que $D\left(x^{(k)}\right)$ converge para $D(x^*)$, sendo x^* uma solução ótima de (2.2). □

Lema 7. *Se a sucessão $\{q_k^b\}$ é infinita, então*

$$\lim_{k\to\infty}(q^* - q_{k+1}^b)(q^* - q_k^b) = 0,$$

isto é, $\{q_k^b\}$ converge superlinearmente para q^*.

Demonstração. Note-se que, para todo o k tal que $q_k^b > q^*$, se tem $q_{k+1}^b = q_k$. Seja $F(q_{k-1}) < 0$ e $F(q_k) < 0$. Por definição de q_k^b,

$$q^* < q_k < q_{k-1} \quad \text{e} \quad q_{k-1} \leq q_k^b.$$

Então, como $q_{i-1} \leq q_k^b$,

$$\begin{aligned}\frac{q_{k+1}^b - q^*}{q_k^b - q^*} &\leq \frac{q_k - q^*}{q_{k-1} - q^*} \\ &\leq \frac{q_k - q^a}{q_{k-1} - q^a}\end{aligned}$$

pois $q^a \leq q^*$,

$$\leq \frac{q_k - q^a}{q^b - q^a}$$

pois $q^b \leq q_{k-1}$ e, por (2.6),

$$= \frac{1}{d(k)}$$

Tendo-se $k \to \infty$, conclui-se que $d(k) \to \infty$ por hipótese sobre $d(k)$. □

Lema 8. *Se q_i pertence a $\{q_k^c\}$, então q_{i-1} pertence a $\{q_k^a\}$ e q_{i+1} pertence a $\{q_k^a\}$ ou $\{q_k^b\}$.*

Demonstração. A demonstração decorre diretamente da definição de $\{q_k^a\}$, $\{q_k^b\}$ e $\{q_k^c\}$. □

O teorema seguinte é consequência dos três lemas anteriores.

Teorema 2. *A sucessão $\{q_i\}$ gerada pelo Algoritmo 3 contém uma subsucessão $\{q_t^1\}$ com as seguintes propriedades.*

1. *Se $q_t^1 = q_i$, então $q_{t+1}^1 = q_{i+1}$ ou $q_{t+1}^1 = q_{t+2}$.*
2. *A sucessão $\{q_t^1\}$ pode ser separada em duas subsucessões, $\{q_k^a\}$ e $\{q_k^b\}$, onde $\{q_k^a\}$ converge superlinearmente para $(q^*)^-$ se é infinita, e $\{q_k^b\}$ converge superlinearmente para $(q^*)^+$ se também é infinita.*

Consequentemente, a sucessão de iterações gerada pelo Algoritmo 3 consiste, essencialmente, em duas subsucessões que convergem superlinearmente. O Teorema 3 mostra que q^* está contido em $\left[q^a, q^b\right]$.

Teorema 3. *Seja $L = q_2 - q_1$ a amplitude do intervalo inicial definido no Algoritmo 3. Após p iterações, o intervalo $\left[q^a, q^b\right]$ satisfaz*

$$q^b - q^a \leq \frac{L}{2^{\lceil \frac{1}{2}p \rceil}}, \tag{2.7}$$

onde $\lceil . \rceil$ representa o maior valor inteiro contido no argumento.

Demonstração. Sejam $\{q_i^a\}$ e $\{q_i^b\}$ os valores de q^a e q^b obtidos na i-ésima iteração do método, respetivamente. Em particular, q_0^a e q_0^b denotam os valores iniciais de q^a e q^b, respetivamente.

Para cada caso, considerem-se as seguintes propriedades.

1. Se $q_i \in \{q_k^b\}$, então $q_i^b - q_i^a \leq \frac{1}{d(k)} \left(q_{i-1}^b - q_{i-1}^a\right)$.
2. Se $q_{i-1} \in \{q_k^a\}$ ou $q_i = q_1$, então $q_i^b - q_i^a \leq \frac{1}{2} \left(q_{i-1}^b - q_{i-1}^a\right)$.
3. Se $q_i \in \{q_k^c\} \cup \{q_k^b\}$ e $q_i \in \{q_k^a\}$, então $q_i^b - q_i^a \leq d(k)(d(k)-1) \left(q_{i-1}^b - q_{i-1}^a\right)$.

Através do Lema 8, esgotam-se todos os casos possíveis. Desde que, por hipótese, o caso 3 e $d(k) \geq 2$ não ocorram em simultâneo, a desigualdade (2.7) é satisfeita.

□

Os Teoremas 2 e 3 mostram que o Algoritmo 3 combina características dos métodos de Newton e procura binária [17].

2.3 Abordagem não paramétrica

Nesta secção descreve-se um método não paramétrico, baseado numa mudança de variável que é válida apenas para o caso linear, em que N e D são funções afins.

Considere-se o programa linear fracionário definido por

$$\begin{array}{ll} \min & R(c) = \dfrac{N(x)}{D(x)} \equiv \dfrac{c^\top x + \alpha}{d^\top x + \beta} \\ \text{s. a} & Ax \leq b \\ & x \geq 0, \end{array} \tag{2.8}$$

com $\alpha, \beta \in \mathbb{R}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ e $c, d \in \mathbb{R}^n$. Tal como na abordagem paramétrica, supõe-se que $d^\top x + \beta > 0$ e que a região admissível, definida por $S \equiv \{x \in \mathbb{R}^n \mid Ax \leq b \wedge x \geq 0\}$, é não vazia e limitada.

Charnes e Cooper [5] propuseram a transformação deste problema num programa linear, por aplicação da mudança de variável

$$y \equiv tx, \text{ com } t \geq 0,$$

onde $d^\top y + \beta t = \gamma$, para algum $\gamma > 0$. Multiplicando o numerador e o denominador da função objetivo em (2.8) por t, obtém-se o problema equivalente

$$\begin{array}{ll} \min & c^\top y + \alpha t \\ \text{s. a} & Ay - bt \leq 0 \\ & d^\top y + \beta t = \gamma \\ & y, t \geq 0 \end{array} \tag{2.9}$$

Este novo problema é um programa linear, que pode ser resolvido através do algoritmo simplex ou de algoritmos de pontos interiores, pelo que, ao contrário dos métodos anteriores, se considera não ser iterativo.

Lema 9. *Qualquer que seja (y,t) satisfazendo as restrições de (2.9), tem-se $t > 0$.*

Demonstração. Sejam x^* e $(y^*, 0)$ soluções de (2.8) e (2.9), respetivamente. Então $x_\mu \equiv x^* + \mu y^*$ é solução de (2.8), para $\mu > 0$ com $Ay^* \leq 0$, $y^* \geq 0$, e assim S não seria limitado, conforme suposto inicialmente. □

Teorema 4. *Sejam x^* e (y^*, t^*) soluções ótimas de (2.8) e (2.9), respetivamente, então $\frac{y^*}{t^*}$ é uma solução ótima de (2.8).*

Demonstração. Por absurdo suponha-se que existe uma solução ótima $x^* \in S$, tal que, para $\frac{y^*}{t^*}$, se tem

$$\frac{c^\top x^* + \alpha}{d^\top x^* + \beta} > \frac{c^\top \left(\frac{y^*}{t^*}\right) + \alpha}{d^\top \left(\frac{y^*}{t^*}\right) + \beta}.$$

Por hipótese, tem-se $d^\top x^* + \beta = \theta\gamma$, para algum $\theta > 0$.

Sejam $\hat{y} = \theta^{-1}x^*$ e $\hat{t} = \theta^{-1}$. Então,

$$\theta^{-1}(d^\top x^* + \beta) = (d^\top \hat{y} + \hat{t}\beta) = \gamma$$

e $(\hat{y}, \hat{t})$ satisfaz $A\hat{y} - b\hat{t} \leq 0$, sendo $\hat{y}, \hat{t} \geq 0$. Mas,

$$\frac{c^\top x^* + \alpha}{d^\top x^* + \beta} = \frac{\theta^{-1}(c^\top x^* + \alpha)}{\theta^{-1}(d^\top x^* + \beta)} = \frac{c^\top \hat{y} + \alpha\hat{t}}{d^\top \hat{y} + \beta\hat{t}} = \frac{c^\top \hat{y} + \alpha\hat{t}}{\gamma},$$

logo,

$$\frac{c^\top \left(\frac{y^*}{t^*}\right) + \alpha}{d^\top \left(\frac{y^*}{t^*}\right) + \beta} = \frac{c^\top y^* + \alpha t^*}{d^\top y^* + \beta t^*} = \frac{c^\top y^* + \alpha t^*}{\gamma}.$$

Além disso,

$$\frac{c^\top x^* + \alpha}{d^\top x^* + \beta} > \frac{c^\top \left(\frac{y^*}{t^*}\right) + \alpha}{d^\top \left(\frac{y^*}{t^*}\right) + \beta}.$$

Então, pela primeira condição do teorema, tem-se $\gamma \neq 0$, logo

$$c^\top \hat{y} + \alpha\hat{t} > c^\top y^* + t^*,$$

o que contraria o facto de (y^*, t^*) ser uma solução ótima de (2.9). □

2.4 Experiência computacional

Na presente secção pretende-se testar a eficiência dos métodos descritos anteriormente para um conjunto de programas lineares fracionários. Para este efeito implementaram-se os seguintes códigos:

- `MCC`: resolução do programa linear resultante da mudança de variável introduzida por Charnes e Cooper;
- `MD`: método de Dinkelbach, Algoritmo 1;
- `MN`: método de Newton;
- `MPB`: método da procura binária, Algoritmo 2;
- `MPBD`: método da procura binária modificado, Algoritmo 3.

Os códigos foram implementados em Matlab 9.5 (R2018b), recorrendo à rotina `cplexlp`, do *solver* CPLEX 12.8, para resolver os programas lineares. Como condição de paragem dos métodos iterativos consideraram-se uma tolerância máxima de $\delta = 10^{-4}$ e até 500 iterações. Além disso, o intervalo escolhido inicialmente para o parâmetro q no código `MPB` foi $[0, 1]$. Os vários códigos foram testados num computador com um processador Intel Core i3 CPU 330 a 1.20GHz.

O pseudo-código apresentado no Algoritmo 4 cria um programa linear fracionário aleatório a partir de quatro parâmetros, o número de variáveis, n, o número de restrições, nr, o limite superior dos

parâmetros do problema, C, e uma semente para iniciar o gerador de números aleatórios. O código define cinco variáveis:

1. $N = [N_j]_{j=1,\ldots,n}$, com N_j os n coeficientes do numerador da função objetivo, gerados de forma uniforme em $\{1,\ldots,C\}$, $j = 1\ldots,n$;

2. $D = [D_j]_{j=1,\ldots,n}$, com D_j os n coeficientes do denominador da função objetivo, igualmente gerados de forma uniforme em $\{1,\ldots,C\}$, $j = 1,\ldots,n$;

3. α gerado de forma uniforme em $\{1,\ldots,C\}$;

4. $A = [A_{ij}]_{i=1,\ldots,nr;j=1,\ldots,n}$, com A_{ij} os $nr \times n$ coeficientes tecnológicos do problema, gerados aleatoriamente em $\{1,\ldots,C\}$;

5. $b = [b_i]_{i=1,\ldots,nr}$, com b_i os nr termos independentes do problema.

Os três primeiros pontos definem a função objetivo, da forma $\frac{Nx}{Dx+\alpha}$, enquanto que os dois últimos definem as restrições propriamente ditas, $Ax \leq b$. Para garantir que o problema tem solução, o vetor b é definido a partir de Ax, para um vetor $x \in [1,C]^n$.

Algoritmo 4: Gerador de programas lineares fracionários

Function `MakeLP`$(n,\ nr,\ C,\ seed)$**:**
  $rng(seed)$
  $N \leftarrow round(C\,rand(1,n))$
  $D \leftarrow round(C\,rand(1,n))$
  $\alpha \leftarrow round(C\,rand(1,1))$
  while $D = N$ **do** $D \leftarrow round(C\,rand(1,n))$
  $A \leftarrow round(-C + 2C\,rand(nr,n))$
  $x \leftarrow C\,rand(1,n)$
  $b \leftarrow round(Ax)$
  return N, D, α, A, b

O conjunto de testes considerados incluiu problemas gerados aleatoriamente com $n = 60, 80, 100$ variáveis e $nr = n \pm 20$ restrições. Utilizaram-se os limites $C = 10, 100$. Para cada dimensão geraram-se e resolveram-se 30 problemas com sementes diferentes.

2.4.1 Método de Newton com q variável

Podendo o método de Newton ser interpretado como uma versão particular do método de Dinkelbach em que o valor inicial do parâmetro q não é necessariamente 0, testou-se o código `MN` para os problemas indicados acima e considerando diferentes valores inciais para q. Nomeadamente, $q \in \left\{\frac{1}{2}, 1, \frac{3}{2}, 2\right\}$.

Os resultados médios em termos de tempo de execução e de número de iterações realizadas são apresentados nas Tabelas 2.1 e 2.2. O melhor tempo médio para cada dimensão encontra-se assinalado a vermelho.

Tabela 2.1 Tempo de execução (em segundos) e número de iterações médios do método de Newton para q variável em programas lineares fracionários, com $C = 10$

	$q = 0.5$			$q = 1.0$			$q = 1.5$			$q = 2.0$		
(n, nr)	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
$(60, 40)$	0.0269	0.0342	7	0.0275	0.0296	7	0.0268	0.0293	7	0.0273	0.0298	7
$(60, 80)$	0.0471	0.0365	5	0.0467	0.0396	5	0.0493	0.0429	5	0.0461	0.0354	5
$(80, 60)$	0.0523	0.0381	7	0.0534	0.0428	7	0.0514	0.0365	7	0.0539	0.0404	7
$(80, 100)$	0.0750	0.3888	5	0.0750	0.0357	5	0.0761	0.0357	5	0.0768	0.0416	5
$(100, 80)$	0.0926	0.0490	7	0.0824	0.0442	7	0.0807	0.0413	7	0.0823	0.0427	7
$(100, 120)$	0.1304	0.0395	5	0.1296	0.0404	5	0.1317	0.0398	5	0.1316	0.0430	5

Tabela 2.2 Tempo de execução (em segundos) e número de iterações médios do método de Newton para q variável em programas lineares fracionários, com $C = 100$

	$q = 0.5$			$q = 1.0$			$q = 1.5$			$q = 2.0$		
(n, nr)	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
$(60, 40)$	0.0308	0.0346	8	0.0299	0.0331	8	0.0298	0.0340	8	0.0329	0.0411	8
$(60, 80)$	0.0480	0.0428	5	0.0481	0.0390	5	0.0491	0.0414	5	0.0510	0.0464	8
$(80, 60)$	0.0516	0.0411	7	0.0669	0.0693	7	0.0564	0.0509	7	0.0517	0.0405	7
$(80, 100)$	0.0766	0.0332	5	0.0843	0.0537	5	0.0770	0.0310	5	0.0871	0.0538	5
$(100, 80)$	0.0889	0.0469	8	0.0889	0.0410	8	0.1036	0.0469	8	0.0980	0.0525	8
$(100, 120)$	0.2035	0.0773	6	0.1501	0.0490	6	0.2022	0.0687	6	0.1358	0.0407	6

Os resultados obtidos evidenciam que os tempos de execução exigidos nos vários casos são muito dependentes da dimensão do problema que se pretende resolver, nomeadamente no caso $C = 10$. Na Tabela 2.2, os menores tempos de execução obtêm-se quando $q = 0.5$.

Em relação ao número de iterações, verifica-se que é praticamente constante para cada dimensão, isto é, que é independente de q.

Concluindo, apesar de os tempos de execução médios registados serem bastante semelhantes, os factores que mais os influenciaram foram a dimensão dos problemas e a relação entre o número de variáveis e o número de restrições.

2.4.2 Métodos paramétricos e não paramétrico

Numa segunda fase testaram-se os restantes métodos implementados para o mesmo conjunto de problemas. Os tempos de execução, os desvio padrão correspondentes e os números de iterações (sempre que aplicáveis) médios exigidos para resolver os problemas são resumidos nas Tabelas 2.3 e 2.4. O melhor tempo médio para cada dimensão é assinalado a negrito e o melhor tempo médio obtido por um método iterativo é assinalado a vermelho. Analogamente, o menor número médio de iterações para cada dimensão encontra-se sublinhado.

Com base nos resultados nas Tabelas 2.3 e 2.4 observa-se que o código `MCC` foi o mais eficiente em todos os casos. Este facto não é surpreedente, uma vez que este método resolve um único programa linear depois de realizada a mudança de variável. Nota-se, contudo, que esta abordagem é menos geral do que as restantes, na medida em que trabalha com variáveis de decisão contínuas e que transforma a região admissível do problema.

Tabela 2.3 Tempo de execução (em segundos) e número de iterações médios em programas lineares fracionários, com $C = 10$

	MCC		MD			MPB			MPBM		
(n,nr)	$\bar{t}$	s_t	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
$(60,40)$	**0.0177**	0.0452	0.0263	0.0314	7	0.6618	0.8748	179	1.3937	0.2041	500
$(60,80)$	**0.0179**	0.0304	0.0468	0.0362	5	0.3391	0.8247	34	3.5691	0.4764	500
$(80,60)$	**0.0147**	0.0297	0.0599	0.0469	7	1.1934	1.6810	147	2.4107	0.2838	500
$(80,100)$	**0.0218**	0.0304	0.0757	0.0414	5	0.7019	1.8467	34	7.1082	3.3648	500
$(100,80)$	**0.0206**	0.0345	0.0809	0.0420	7	1.7069	3.2233	98	4.8959	1.2550	500
$(100,120)$	**0.0303**	0.0311	0.1428	0.0543	5	0.7229	0.2862	19	11.2058	4.5190	500

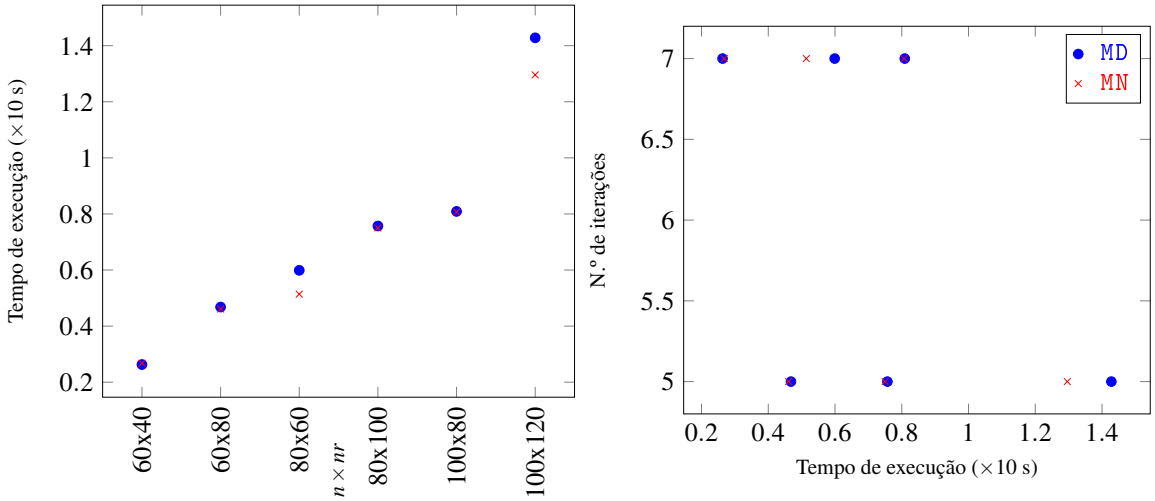
Tabela 2.4 Tempo de execução (em segundos) e número de iterações médios em programas lineares fracionários, com $C = 100$

	MCC		MD			MPB			MPBM		
(n,nr)	$\bar{t}$	s_t	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
$(60,40)$	**0.0126**	0.0392	0.0304	0.0362	8	0.7710	0.9171	199	1.4224	0.1584	500
$(60,80)$	**0.0215**	0.0381	0.0483	0.0368	5	0.5866	1.3918	56	3.7074	0.5068	500
$(80,60)$	**0.0170**	0.0363	0.0514	0.0398	7	0.8485	1.1660	152	3.2609	0.7369	500
$(80,100)$	**0.0246**	0.0324	0.0807	0.0343	5	0.8987	1.8552	40	8.0057	3.0078	500
$(100,80)$	**0.0202**	0.0276	0.0860	0.0429	8	1.6499	2.3425	152	5.0479	0.6334	500
$(100,120)$	**0.0392**	0.0398	0.1331	0.0410	6	1.0624	1.8485	41	13.4619	6.3334	500

No caso dos métodos iterativos, o mais eficiente, em termos de tempos de execução médios, foi consistentemente o código `MD`, atingindo também o número de iterações mais baixo. Os valores de desvio padrão registados são relativamente pequenos, o que mostra alguma estabilidade de comportamento dos métodos para o conjunto de testes considerado. Além disto, é evidente a diminuição do número de iterações realizadas nos casos em que o número de restrições é superior ao número de variáveis. Os resultados do código `MPBM` indicam que foram sempre atingidas 500 iterações, ou seja, o número máximo de iterações imposto. Apesar de se terem utilizado os mesmo critérios de paragem para todos os métodos implementados, nestes casos a solução ótima foi encontrada mas não obedece ao primeiro critério imposto (ou seja, o valor da função objetivo ultrapassa a tolerância permitida). Esta questão poderia ser resolvida relaxando a tolerância utilizada, ou acrescentando uma nova condição de paragem, que comparasse as aproximações da solução obtidas em iterações consecutivas da solução.

Nas Figuras 2.1 e 2.2 reúnem-se os resultados dos dois métodos iterativos que apresentaram o melhor comportamento para os problemas testados `MN` e `MD`. Para este efeito selecionaram-se os resultados obtidos pelo `MN` correspondentes à melhor aproximação inicial q. Verifica-se que o código `MN` apenas foi o mais eficiente em dois casos, sendo semelhante ao `MD` nos restantes. No caso do `MPBM`, o número máximo de iterações é atingido em todas as dimensões, levando a concluir que a solução ótima não chega a ser atingida dentro da tolerância pretendida. Deste modo, o método torna-se menos eficaz na resolução de problemas lineares fracionários genéricos.

Em conclusão, sublinha-se novamente que a abordagem proposta por Charnes e Cooper mostrou-se sempre consideravelmente mais rápida do que os restantes métodos. Contudo, estes últimos são mais flexíveis, uma vez que dependem apenas da resolução de um problema com uma estrutura semelhante à original, mas com uma função objetivo mais simples. Se o problema inicial é linear

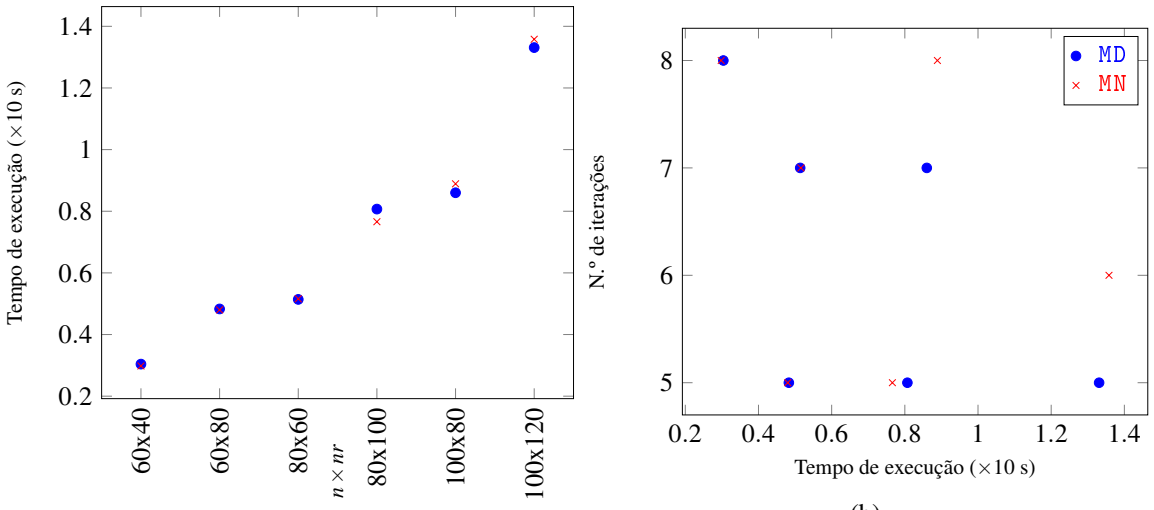


(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

Figura 2.1 Resultados dos métodos `MD` e `MN` para programas lineares fracionários, com $C = 10$

fracionário, os vários subproblemas são versões lineares do problema dado. Isto é especialmente útil em problemas combinatórios com estruturas particulares e que possam ser resolvidos por métodos dedicados e eficientes, como os problemas da afetação ou da mochila (a analisar no Capítulo 4).



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

Figura 2.2 Resultados dos métodos `MD` e `MN` para programas lineares fracionários, com $C = 100$

Capítulo 3

Método biobjetivo para programação linear fracionária

O problema linear fracionário formulado em (2.8) envolve duas funções afins, N e D. No presente capítulo este problema é interpretado como um problema linear com duas funções objetivo. O capítulo inicia-se com a introdução de alguns conceitos de otimização multiobjetivo e prossegue com a adaptação de um método clássico de otimização multiobjetivo à resolução do problema (2.8), seguido do seu melhoramento quando conjugado com resultados apresentados no Capítulo 2. Por fim, apresentam-se resultados de testes computacionais para os dois métodos descritos, quando aplicados a programas lineares fracionários gerados de forma aleatória.

3.1 Otimização multiobjetivo

Considere-se o problema de otimização multiobjetivo

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & x \in S \end{array} \tag{3.1}$$

onde $S \subseteq \mathbb{R}^n$ e $f(x) = (f_1(x), \ldots, f_k(x)) \in \mathbb{R}^k$. Uma vez que f toma valores em $\mathbb{R}^k$, este problema depende de diferentes funções objetivo, $f_1(x), \ldots, f_k(x)$, por vezes contraditórias. A título de exemplo considere-se um caso biobjetivo, $k = 2$, em que se pretende maximizar o desempenho, mas minimizar o custo a ele associado. Em geral, estes dois aspetos podem ser conflituosos, pois a minimização de um deles pode não coincidir com a do outro [12]. O mesmo acontece em problemas com mais funções objetivo, a maioria dos quais não tem uma solução que seja simultaneamente ótima para todas as funções. Alternativamente, em otimização multiobjetivo pretende-se encontrar soluções que conciliem as várias funções objetivo, procurando-se determinar as soluções que melhoram uma delas sem prejudicar nenhuma outra.

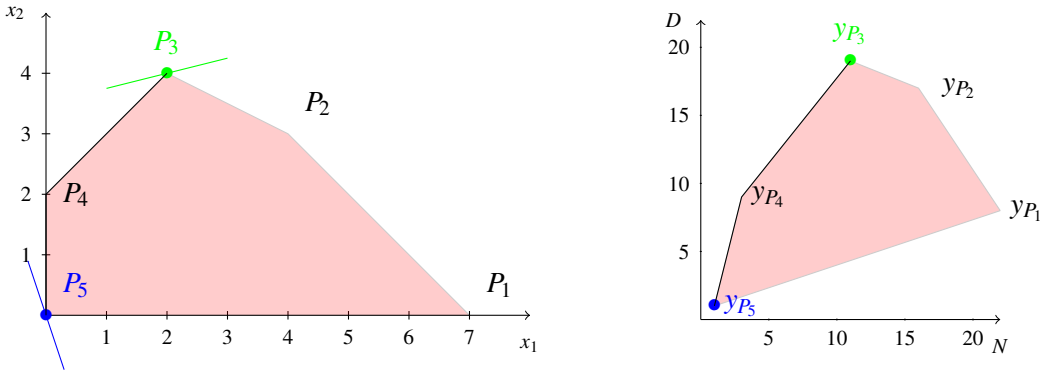
Definição 1. *[12] Seja $\hat{x} \in S$ uma solução admissível do problema (3.1). Esta solução diz-se uma solução eficiente, ou uma solução ótima de Pareto, se não existe nenhuma outra solução $x \in S$ que satisfaça $f(x) \leq f(\hat{x})$ (isto é, $f_i(x) \leq f_i(\hat{x})$, para qualquer $i = 1, \ldots, k$) e $f(x) \neq f(\hat{x})$. Se $\hat{x}$ é uma solução eficiente, então $f(\hat{x})$ diz-se um ponto não dominado.*

O conjunto de todas as soluções eficientes $\hat{x} \in S$ denomina-se por conjunto de eficiência e denota-se por S_E. O conjunto de todos os pontos não dominados $\hat{y} = f(\hat{x})$, onde $\hat{x} \in S_E$, denomina-se conjunto não dominado ou fronteira de Pareto e denota-se por S_{ND}.

Tome-se como exemplo (adaptado de [7], p. 150) o programa linear com duas funções objetivo definido por

$$\begin{aligned}
\min \quad & N(x) \equiv 3x_1 + x_2 + 1 && \text{(3.2a)}\\
\max \quad & D(x) \equiv x_1 + 4x_2 + 1 && \text{(3.2b)}\\
\text{s. a} \quad & -x_1 + x_2 \leq 2 && \text{(3.2c)}\\
& x_1 + x_2 \leq 7 && \text{(3.2d)}\\
& x_1 + 2x_2 \leq 10 && \text{(3.2e)}\\
& x_1, x_2 \geq 0 && \text{(3.2f)}
\end{aligned}$$

A região sombreada na Figura 3.1a ilustra a região admissível deste programa linear. As linhas azul e verde representam uma das retas definida pelas funções objetivo N e D, respetivamente, que intersetam os pontos ótimos relativamente a cada uma. O ponto P_5, a azul, é solução ótima relativamente à função N, enquanto que o ponto P_3, a verde, é o ótimo relativamente a D. Os restantes extremos da região admissível, assim como as imagens correspondentes, $y_P \equiv (N(P), D(P))$, estão listados na Tabela 3.1.



(a)
Região admissível

(b)
Espaço das funções objetivo

Figura 3.1 Regiões de soluções e imagens do problema (3.2)

A Figura 3.1b mostra a região das imagens das soluções admissíveis do problema (3.2). Nota-se que os pontos y_{P_3}, y_{P_4} e y_{P_5} não são dominados por nenhum outro, portanto correspondem a soluções eficientes. O mesmo acontece com as soluções nos segmentos de reta $[P_3, P_4]$ e $[P_4, P_5]$. Qualquer outra solução admissível do problema é dominada por alguma destas, logo não é eficiente.

Tabela 3.1 Pontos extremos nas Figuras 3.1a e 3.1b

	P_1	P_2	P_3	P_4	P_4
P	$(7,0)$	$(4,3)$	$(2,4)$	$(0,2)$	$(0,0)$
y_P	$(22,8)$	$(16,17)$	$(11,19)$	$(3,9)$	$(1,1)$

Em programação linear multiobjetivo inteira é necessário distinguir dois tipos de soluções eficientes: as suportadas e as não suportadas. Este último conceito não tem utilidade quando as variáveis de decisão são contínuas.

Definição 2. *[12] Uma solução eficiente $\hat{x} \in S_E$ do problema (3.1) diz-se suportada se é solução ótima de um problema mono-objetivo em que a função objetivo resulta de uma soma ponderada das funções objetivo do problema. Caso contrário, $\hat{x}$ diz-se uma solução eficiente não suportada.*

Dois métodos clássicos para resolver problemas de otimização multiobjetivo são o método ε-restringido e o método da soma ponderada [7]. Em ambos os casos, é resolvida uma sequência de problemas mono-objetivo. No primeiro caso, reduz-se o problema multiobjetivo a um problema mono-objetivo, fixando uma função objetivo e considerando que as restantes são limitadas por constantes escolhidas de modo adequado. O programa a resolver é

$$\begin{array}{ll} \min & f_i(x) \\ \text{s. a} & f_j \leq \varepsilon_j, \text{ para } j = 1,\ldots,k,\ j \neq i \\ & x \in S, \end{array}$$

onde $\varepsilon \in \mathbb{R}^k$ é um vetor constante escolhido de forma conveniente. Resolver este último problema para diferentes vetores ε permite encontrar várias soluções eficientes do problema original. No segundo caso, repete-se a resolução do programa

$$\begin{array}{ll} \min & \displaystyle\sum_{i=1}^{k} q_i f_i(x) \\ \text{s. a} & x \in S, \end{array}$$

para diferentes vetores $q \in \mathbb{R}^k$, tais que $\sum_{i=1}^{k} q_i = 1$ e $q_i > 0$ para $i = 1,\ldots,k$. Quando utilizado pontualmente o vetor q reflete a importância relativa de cada objetivo. Variar este vetor de forma sistemática, conduz à fronteira de Pareto do problema (3.1). Tal como na abordagem anterior, este problema tem uma única função objetivo e pressupõe-se ser de fácil resolução.

Uma vez que será utilizado adiante, descreve-se em seguida um método, proposto por Cohon [8], para atualizar de forma automática o vetor q a utilizar no problema biobjetivo

$$\begin{array}{ll} \min & N(x) \\ \max & D(x) \\ \text{s. a} & x \in S, \end{array} \tag{3.3}$$

Como problema auxiliar é utilizado

$$\begin{array}{ll} \min & W(x) \equiv q_1 N(x) - q_2 D(x) \\ \text{s. a} & x \in S, \end{array} \tag{3.4}$$

onde W é a função de soma ponderada a otimizar e que depende das ponderações definidas por

$$q_1 = D(B) - D(A) \ \text{ e } \ q_2 = N(B) - N(A), \tag{3.5}$$

onde A e B são soluções eficientes conhecidas, tais que $N(A) < N(B)$ e $D(A) < D(B)$.

O método introduzido por Cohon é resumido no Algoritmo 5, onde y_A e y_B denotam $(N(A), D(A))$ e $(N(B), D(B))$, respetivamente. Este método determina as soluções eficientes suportadas do problema (3.3) e é também conhecido por método *non-inferior set estimation* (NISE). O nome deve-se ao facto de permitir encontrar apenas uma aproximação para a fronteira de Pareto, quando interrompido antes de a lista X se tornar vazia.

Algoritmo 5: Método da soma ponderada para programas lineares biobjetivo

$A \leftarrow$ solução eficiente de $\min\{N(x) \mid x \in S\}$
$B \leftarrow$ solução eficiente de $\max\{D(x) \mid x \in S\}$
$X \leftarrow \{(A,B)\}$
$S_E \leftarrow \{A,B\}$
while $X \neq \emptyset$ **do**
    $(A,B) \leftarrow$ elemento do conjunto X
    $X \leftarrow X - \{(A,B)\}$
    $q_1 \leftarrow D(B) - D(A)$
    $q_2 \leftarrow N(B) - N(A)$
    $C \leftarrow$ solução ótima de $\min\{W(x) \mid x \in S\}$
    if $W(C) \neq W(A)$ *and* $W(C) \neq W(B)$ **then**
        $X \leftarrow X \cup \{(A,C),(C,B)\}$
        $S_E \leftarrow S_E \cup \{C\}$
return S_E

Interessa sublinhar que não é suficiente otimizar N ou D isoladamente para garantir que as soluções de partida do Algoritmo 5, A ou B, são eficientes, uma vez que qualquer destas soluções pode ter ótimos alternativos. Isto significa que A deve ser o minimizante de N para o qual D é máximo, ou seja, a solução lexicograficamente ótima relativamente a (N,D). Analogamente, B deve ser o maximizante de D para o qual N é mínimo, isto é, a solução lexicograficamente ótima relativamente a (D,N). A solução A pode calcular-se começando por resolver o problema

$$\begin{aligned} \min \quad & N(x) \\ \text{s. a} \quad & x \in S \end{aligned}$$

seguido de

$$\begin{aligned} \max \quad & D(x) \\ \text{s. a} \quad & x \in S \\ & N(x) \leq y^*, \end{aligned}$$

onde y^* é o valor objetivo ótimo do primeiro problema. A solução do segundo problema é A. O ponto B pode ser obtido de modo análogo.

O número de iterações efetuadas pelo Algoritmo 5 é proporcional ao número de soluções não dominadas do problema a resolver.

3.2 Método da soma ponderada para programas lineares fracionários

Considere-se novamente um problema linear fracionário da forma

$$\begin{array}{ll} \min & \dfrac{N(x)}{D(x)} \\ \text{s. a} & x \in S, \end{array} \tag{3.6}$$

onde N e D são funções afins. Este programa envolve o quociente de duas funções e, portanto, está relacionado com o programa linear biobjetivo

$$\begin{array}{ll} \min & N(x) \\ \max & D(x) \\ \text{s. a} & x \in S \end{array} \tag{3.7}$$

na medida em que a solução ótima do primeiro problema deve ter valores pequenos de N e grandes de D. Esta é uma interpretação intuitiva do problema, contudo, em seguida, mostra-se que a solução ótima do problema (3.6) é também uma solução eficiente suportada do problema biobjetivo correspondente, (3.7).

Teorema 5. *Sejam N e D tais que $N(x), D(x) > 0$, para qualquer $x \in S$.*

1. *Uma solução ótima do problema linear fracionário (3.6) é solução eficiente do problema biobjetivo (3.7).*

2. *Uma solução ótima do problema*

$$\begin{array}{ll} \max & N(x)/D(x) \\ s.\ a & x \in S \end{array} \tag{3.8}$$

é solução eficiente do problema

$$\begin{array}{ll} \max & N(x) \\ \min & D(x) \\ s.\ a & x \in S. \end{array} \tag{3.9}$$

Demonstração. Mostra-se apenas o primeiro resultado, uma vez que o segundo se pode verificar de forma análoga.

Procede-se por contradição, considerando uma solução ótima de (3.6), x^*, e supondo que esta é solução dominada de (3.7).

- Se x^* é solução admissível do problema (3.6), então tem-se $x^* \in S$, o que significa que x^* também é solução admissível de (3.7).

- Suponhamos que existe uma solução $\hat{x} \in S$ que domina x^*. Então,

$$N(\hat{x}) \leq N(x^*) \ \text{ e } \ D(\hat{x}) \geq D(x^*),$$

donde, atendendo a que $N(x), D(x) > 0$, se tem

$$\frac{N(\hat{x})}{D(\hat{x})} \leq \frac{N(x^*)}{D(\hat{x})} \leq \frac{N(x^*)}{D(x^*)}.$$

Além disto, como uma das desigualdades anteriores é satisfeita de modo estrito, isto é,

$$N(\hat{x}) < N(x^*) \text{ ou } D(\hat{x}) > D(x^*),$$

conclui-se que

$$\frac{N(\hat{x})}{D(\hat{x})} < \frac{N(x^*)}{D(x^*)},$$

pelo que x^* não seria solução ótima de (3.6), como suposto inicialmente.

□

Esta demonstração continua válida para programas lineares com variáveis inteiras. O resultado seguinte conclui que, ao contrário do que se passa em otimização linear biobjetivo inteira, neste caso, as soluções eficientes não suportadas não são relevantes. Isto é particularmente importante pelo facto de o cálculo de soluções eficientes não suportadas ser consideravelmente mais difícil do que a determinação de soluções suportadas, portanto este resultado tem consequências práticas bem concretas.

Lema 10. *1. Existe pelo menos uma solução eficiente suportada do problema (3.7) que é solução ótima do problema (3.6).*

2. Existe pelo menos uma solução eficiente suportada do problema (3.9) que é solução ótima do problema (3.8).

Demonstração. 1. De acordo com o Teorema 1, a solução ótima de

$$\min\{N(x) - q_0 D(x) \mid x \in S\} \tag{3.10}$$

é solução ótima de (3.6) para um parâmetro $q_0 \in \mathbb{R}$ bem escolhido, donde é suficiente escolher as ponderações $q_1 = 1$ e $q_2 = q_0$. Se for necessário impôr que $q_1, q_2 > 0$ e $q_1 + q_2 = 1$, podem considerar-se os parâmetros normalizados

$$q_1 = \frac{1}{q_0 + 1} \text{ e } q_2 = \frac{q_0}{q_0 + 1}.$$

2. Este ponto pode verificar-se de forma análoga ao anterior, atendendo a que a solução ótima de

$$\max\{N(x) - q_0 D(x) \mid x \in S\}$$

é solução ótima de (3.8) para um determinado $q_0 \in \mathbb{R}$, donde decorre o resultado.

□

Considere-se novamente como exemplo o problema linear fracionário (3.2), a partir do qual se pode definir o problema biobjetivo

$$\begin{array}{ll} \min & N(x) \equiv 3x_1 + x_2 + 1 \\ \max & D(x) \equiv x_1 + 4x_2 + 1 \\ \text{s. a} & (3.2c) - (3.2f). \end{array} \tag{3.11}$$

A Tabela 3.2 lista as soluções eficientes e os pontos não dominados deste problema. A última linha da tabela mostra o valor do quociente N/D para cada uma delas. O ponto $P_3 \equiv (0,2)$ é solução suportada do problema, uma vez que minimiza $W(x) = q_1 N(x) + q_2 D(x)$ para

$$q_1 = D(P_3) - D(P_5) = 18 \;\text{ e }\; q_2 = N(P_3) - N(P_5) = 10.$$

De acordo com o Teorema 5 e o Lema 10, conclui-se que $P_4 \equiv (2,0)$ é a solução ótima do problema (3.2).

Tabela 3.2 Soluções eficientes e não dominadas do problema (3.11)

	P_3	P_4	P_5
P	$(2,4)$	$(0,2)$	$(0,0)$
y_P	$(11,19)$	$(3,9)$	$(1,1)$
$N(P)/D(P)$	$11/19$	$1/3$	1

Com base nos resultados anteriores, a solução do problema linear fracionário pode ser obtida aplicando o método da soma ponderada, resumido no Algoritmo 5, seguido da selecão da solução com melhor valor ótimo da função objetivo. Em alternativa, é possível atualizar a melhor solução do problema fracionário à medida que são calculadas novas soluções eficientes. Por completude este método é resumido no Algoritmo 6. Tal como anteriormente, A e B denotam as soluções lexicograficamente ótimas relativamente a (N,D) e a (D,N), respetivamente.

Ao aplicar o Algoritmo 6 ao problema (3.2) começa-se por encontrar os ótimos relativamente às funções N e D, $A \equiv (0,0)$ e $B \equiv (2,4)$, respetivamente, escolhendo-se $Best \equiv B$. Como referido anteriormente, as ponderações definidas a partir deste par de soluções são $q_1 = 18$ e $q_2 = 10$, que levam à função objetivo

$$W(x) = 18N(x) - 10D(x) = 44x_1 - 22x_2 + 8.$$

O ponto da região admissível que minimiza esta função é $C = (0,2)$ e, então, a melhor solução é atualizada com $Best \equiv C$. Em seguida são resolvidos outros dois problemas:

1. um em que se consideram as ponderações $q_1 = 10$ e $q_2 = 8$, que levam à função objetivo

$$W(x) = 10N(x) - 8D(x) = 22x_1 - 22x_2 + 2;$$

2. e outro em que $q_1 = 8$ e $q_2 = 2$, que levam à função objetivo

$$W(x) = 8N(x) - 2D(x) = 22x_1 + 6.$$

Algoritmo 6: Método da soma ponderada para programas lineares fracionários

$A \leftarrow$ solução eficiente de $\min\{N(x) \mid x \in S\}$
$B \leftarrow$ solução eficiente de $\max\{D(x) \mid x \in S\}$
$X \leftarrow \{(A,B)\}$
if $N(A)/D(A) < N(B)/D(B)$ **then** $Best \leftarrow A$
else $Best \leftarrow B$
while $X \neq \emptyset$ **do**
    $(A,B) \leftarrow$ elemento do conjunto X
    $X \leftarrow X - \{(A,B)\}$
    $q_1 \leftarrow D(B) - D(A)$
    $q_2 \leftarrow N(B) - N(A)$
    $C \leftarrow$ solução ótima de $\min\{W(x) \mid x \in S\}$
    if $W(C) \neq W(A)$ *and* $W(C) \neq W(B)$ **then**
        $X \leftarrow X \cup \{(A,C),(C,B)\}$
        if $N(C)/D(C) < N(Best)/D(Best)$ **then** $Best \leftarrow C$
return $Best$

Pode verificar-se que ambos os problemas têm múltiplas soluções ótimas. No primeiro caso todos os pontos do segmento de reta $[A,C]$, e os pontos do segmento de reta $[C,B]$ no segundo. Como a função objetivo W tem igual valor para todos eles, nenhuma nova solução é considerada e o algoritmo termina, apresentando $Best \equiv C$ como resultado.

A título de curiosidade, a rotina `gamultiobj` do *software* Matlab permite obter uma aproximação para o conjunto das soluções eficientes de programas lineares com várias funções objetivo. De acordo com a informação no respetivo manual, esta rotina baseia-se num algoritmo genético. O resultado da aplicação desta rotina ao problema (3.11) está ilustrado na Figura 3.2. Observa-se que o conjunto de soluções encontradas não é completo e que, em particular, não contém a solução ótima procurada, pelo que este método não é útil para resolver programas lineares fracionários. Neste caso, também é possível definir uma tolerância máxima, sendo que o algoritmo é interrompido quando a diferença entre os valores de duas iterações sucessivas é menor ou igual a essa mesma tolerância (em todas as funções objetivo). Em relação ao número de iterações e ao tempo de execução, pode-se definir limites que mais uma vez interrompem o algoritmo. Através de todas estas opções, consegue-se aperfeiçoar a fronteira de Pareto pretendida, tornando-a o mais detalhada possível.

3.3 Método da soma ponderada modificado para programas lineares fracionários

Na secção anterior analisou-se a possibilidade de determinar uma solução ótima do problema linear fracionário (3.6) utilizando um método de soma ponderada. Em seguida, verifica-se que o Algoritmo 6 pode ser simplificado, atendendo a propriedades da função soma ponderada que foram estudadas no Capítulo 2.

x	$(N(x), D(x))$
(-0.0010, -0.0010)	(0.9960, 0.9950)
(0.0058, 0.3921)	(1.4095, 2.5742)
(0.0114, 1.0302)	(2.0644, 5.1322)
(0.0127, 1.4110)	(2.4491, 6.6567)
(0.1428, 1.5332)	(2.9616, 7.2756)
(0.1460, 2.0925)	(3.5305, 9.5160)
(0.2678, 1.5488)	(3.3522, 7.4630)
(0.3207, 2.1736)	(4.1357, 10.0151)
(0.3961, 2.3487)	(4.5370, 10.7909)
(0.9217, 2.7717)	(6.5368, 13.0085)
(1.1236, 2.8910)	(7.2618, 13.6876)
(1.2231, 3.1283)	(7.7976, 14.7363)
(1.3563, 3.3286)	(8.3975, 15.6707)
(1.5148, 3.4528)	(8.9972, 16.3260)
(1.5845, 3.5706)	(9.3241, 16.8669)
(1.7592, 3.6574)	(9.9350, 17.3888)
(2.0272, 3.9797)	(11.0613, 18.9460)

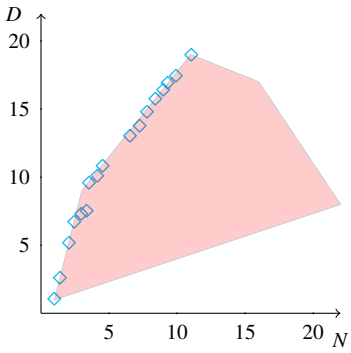


Figura 3.2 Resultado da aplicação da rotina `gamultiobj` ao problema (3.11)

Como referido anteriormente, o problema (3.4) é equivalente a

$$\begin{aligned} \min \quad & Z(x) \equiv N(x) - qD(x) \\ \text{s. a} \quad & x \in S \end{aligned}$$

sendo $q = q_2/q_1$, com q_1 e q_2 os parâmetros definidos em (3.5). A formulação deste problema é semelhante à apresentada em (3.10), sendo A e B definidos novamente como para o Algoritmo 6. Neste método cada par de soluções determinadas origina a resolução de dois novos problemas, o que pode implicar um elevado número de iterações do ciclo `while` na linha 6. Contudo, no Capítulo 2 mostrou-se que a função dependente do parâmetro q_0, definida por (3.10) é monótona. Então, se C é solução ótima de (3.4), definida a partir das soluções A e B, a solução procurada encontra-se entre o par de soluções para o qual a função F muda de sinal. Portanto, é possível reduzir os dois novos pares de soluções considerados em cada iteração do Algoritmo 6, a um único:

- o par (A,C), se $F(A)F(C) < 0$,
- ou o par (C,B), se $F(B)F(C) < 0$.

Este resultado é verificado com mais detalhe no Teorema 6.

O novo método, daqui em diante designado método da soma ponderada modificado, é resumido no Algoritmo 7, mantendo-se a notação do método anterior.

Teorema 6. *O Algoritmo 7 determina uma solução ótima de (3.7).*

Demonstração. De acordo com o Teorema 1,

$$q_0 = \min \left\{ \frac{N(x)}{D(x)} \mid x \in S \right\}$$

Algoritmo 7: Método da soma ponderada modificado para programas lineares fracionários

```
1  A ← solução eficiente de min{N(x) | x ∈ S}
2  B ← solução eficiente de max{D(x) | x ∈ S}
3  X ← {(A,B)}
4  if N(A)/D(A) < N(B)/D(B) then Best ← A
5  else Best ← B
6  while X ≠ ∅ do
7      (A,B) ← elemento do conjunto X
8      X ← X − {(A,B)}
9      q ← (N(B) − N(A)) / (D(B) − D(A))
10     C ← solução ótima de min{Z(x) | x ∈ S}
11     if Z(C) ≠ Z(A) and Z(C) ≠ Z(B) then
12         if Z(A)Z(C) < 0 then
13             B ← C
14             X ← X ∪ {(A,C)}
15         else
16             A ← C
17             X ← X ∪ {(C,B)}
18         if N(C)/D(C) < N(Best)/D(Best) then Best ← C
19 return Best
```

se e só se $F(q_0) = 0$, com $F(q) = \min\{N(x) - qD(x) \mid x \in S\}$. No início do Algoritmo 7 calculam-se A e B, soluções lexicograficamente ótimas relativamente a (N,D), e relativamente a (D,N), respetivamente.

Então,

- A é minimizante de $F(0) = \{N(x) \mid x \in S\}$ e
- B é minimizante de $F(q') = \{N(x) - q'D(x) \mid x \in S\}$ para $q' \in \mathbb{R}$ arbitrariamente grande,

pelo que $F(0) \geq 0$ e $F(q') \leq 0$.

Sem perda de generalidade, suponha-se que $F(0)F(q') < 0$, donde $q_0 \in]0,q'[$ (caso contrário, encontrou-se uma solução ótima do problema e o algoritmo termina). Mostrar-se-á que no final da primeira iteração do ciclo `while` na linha 9 do Algoritmo 7 se continua a ter $q_0 \in [q_A, q_B]$, sendo $F(q_A)F(q_B) \leq 0$, com q_1 e q_2 os parâmetros correspondentes a duas soluções. Nesta iteração é determinada C, solução ótima de

$$\min F(q) = \{N(x) - qD(x) \mid x \in S\}$$

para $q = \dfrac{N(B) - N(A)}{D(B) - D(A)}$. Nas linhas 13 a 16 procede-se à atualização das soluções A e B, por forma a que

1. $F(q_B) = 0$, caso em que o ciclo termina e B é solução ótima de (3.6);
2. $F(q_A)F(q_B) < 0$, donde $q_0 \in]q_A, q_B[$.

Além disto, $q_A \leq q \leq q_B$, pelo que o intervalo encontrado no final da primeira iteração está contido no inicial.

Repetindo este raciocínio, recorrendo ao Teorema 5 e observando que o número de soluções eficientes suportadas do problema é finito, conclui-se que o Algoritmo 7 termina após um número finito de iterações, quando $F(q_B) > 0$, o que significa que B é solução ótima de (3.7). □

Este teorema é o suporte para as linhas 13 a 16 do Algoritmo 7 e justifica que um dos subproblemas que este resolve dê origem a, no máximo, um novo subproblema para resolver. Assim, a demonstração apresentada fornece um limite superior para o número de iterações do Algoritmo 7, conforme enunciado no corolário seguinte.

Corolário 1. *Seja M o número de soluções eficientes suportadas do problema (3.2). O ciclo **while** do Algoritmo 7 é executado $O(\log M)$ vezes.*

O início da aplicação do Algoritmo 7 ao exemplo anterior, problema (3.2), é semelhante ao do Algoritmo 6. Calculam-se os ótimos $A \equiv (0,0)$ e $B \equiv (2,4)$, e fixa-se $Best \equiv B$. Na primeira iteração do ciclo **while**, determina-se $C \equiv (0,2)$, que é a solução admissível que minimiza a função

$$Z(x) = N(x) - \frac{5}{9}D(x) = \frac{22}{9}x_1 - \frac{11}{9}x_2 + \frac{4}{9}$$

para $q = \frac{10}{18}$. Como

$$Z(A) = Z(B) = \frac{4}{9} \text{ e } Z(C) = -2,$$

conclui-se que o próximo par de soluções a considerar é (A,C) e a melhor solução é atualizada com $Best \equiv C$. Em seguida é resolvido o problema de minimização quando se considera o parâmetro $q = \frac{2}{8}$, que leva à função objetivo

$$Z(x) = N(x) - \frac{1}{4}D(x) = \frac{11}{4}x_1 + \frac{3}{4}.$$

Uma vez que este não produz nenhuma nova solução, o algoritmo termina com o resultado $Best \equiv C$.

3.4 Experiência computacional

Para avaliar o comportamento dos dois métodos introduzidos neste capítulo, apresentam-se em seguida os resultados de testes computacionais, baseados nos códigos:

- `MSP`: método da soma ponderada para programas lineares fracionários, Algoritmo 6;
- `MSPM`: método da soma ponderada modificado para programas lineares fracionários, Algoritmo 7.

Tal como no Capítulo 2, estes códigos foram implementados em Matlab 9.5 (R2018b) e os programas lineares foram resolvidos através da rotina `cplexlp`. Os testes foram executados na máquina utilizada anteriormente e o conjunto de problemas de teste foi também o utilizado na Secção 2.4. Os resultados médios relativamente ao tempo de execução e ao número de iterações para 30 problemas de cada dimensão são apresentados nas Tabelas 3.3 e 3.4 e nas Figuras 3.3 e 3.4.

De acordo com os resultados obtidos, conclui-se que o Algoritmo 7 foi bastante mais eficaz do que o Algoritmo 6, devido a uma significativa diminuição do número médio de iterações realizadas e

Tabela 3.3 Tempo de execução (em segundos) e número de iterações médios em programas lineares fracionários, com $C = 10$

	MSP			MSPM		
(n, nr)	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
$(60, 40)$	0.1396	0.0857	40	0.0701	0.0721	22
$(60, 80)$	0.1609	0.1449	25	0.0902	0.1361	12
$(80, 60)$	0.2091	0.1344	37	0.1385	0.1312	24
$(80, 100)$	0.3483	0.2787	31	0.1745	0.2581	13
$(100, 80)$	0.4368	0.1932	43	0.2488	0.2298	26
$(100, 120)$	0.5835	0.3933	34	0.2104	0.3127	11

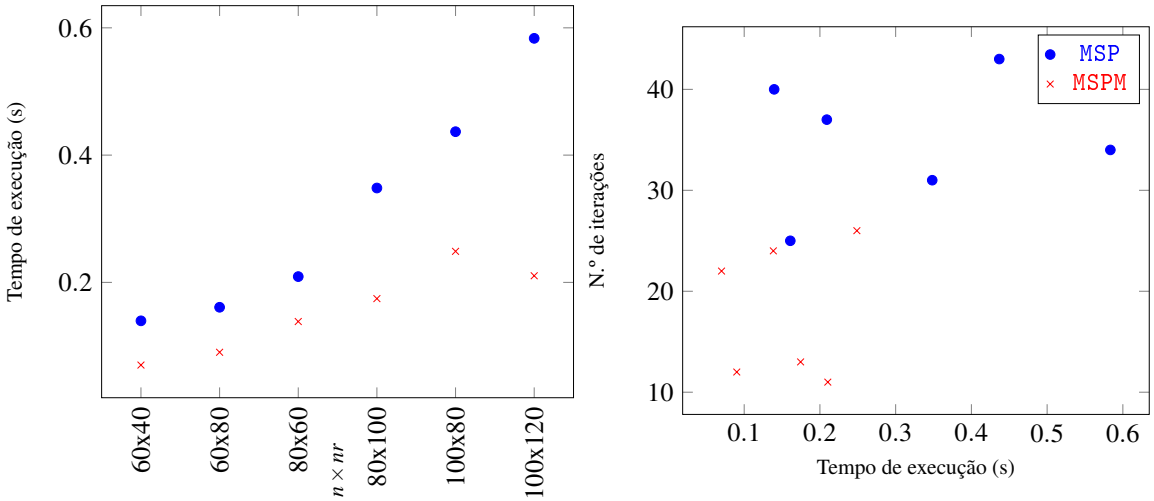
Tabela 3.4 Tempo de execução (em segundos) e número de iterações médios em programas lineares fracionários, com $C = 100$

	MSP			MSPM		
(n, nr)	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
$(60, 40)$	0.1556	0.0945	41	0.0762	0.0789	22
$(60, 80)$	0.2025	0.1604	29	0.0951	0.1273	13
$(80, 60)$	0.2133	0.1210	37	0.1648	0.1591	27
$(80, 100)$	0.3715	0.2718	32	0.3133	0.3097	23
$(100, 80)$	0.3681	0.2194	37	0.2958	0.2516	28
$(100, 120)$	0.6423	0.4353	34	0.5278	0.6802	20

à consequente redução do tempo de execução. Tal como se observou para os métodos iterativos do Capítulo 2, o desvio padrão dos tempos médios de execução é também bastante reduzido para ambos os métodos. Nota-se ainda que o número de iterações para cada dimensão, relacionado com o número de soluções eficientes suportadas determinadas, foi pouco sensível à variação no limite dos custos considerados.

Apesar de as Figuras 3.3a e 3.4a registarem resultados para poucas dimensões de problemas, observa-se uma tendência para um crescimento próximo de exponencial do tempo médio de execução do código `MSP` relativamente à dimensão dos problemas, por oposição a um comportamento praticamente linear dos mesmos resultados para o código `MSPM`. Este facto é mais evidente fixando a atenção apenas nos problemas com $n - 20$, ou $n + 20$, restrições, separadamente, e a diferença parece ser menos marcada para os problemas com maior variação dos custos, ou seja, quando $C = 100$.

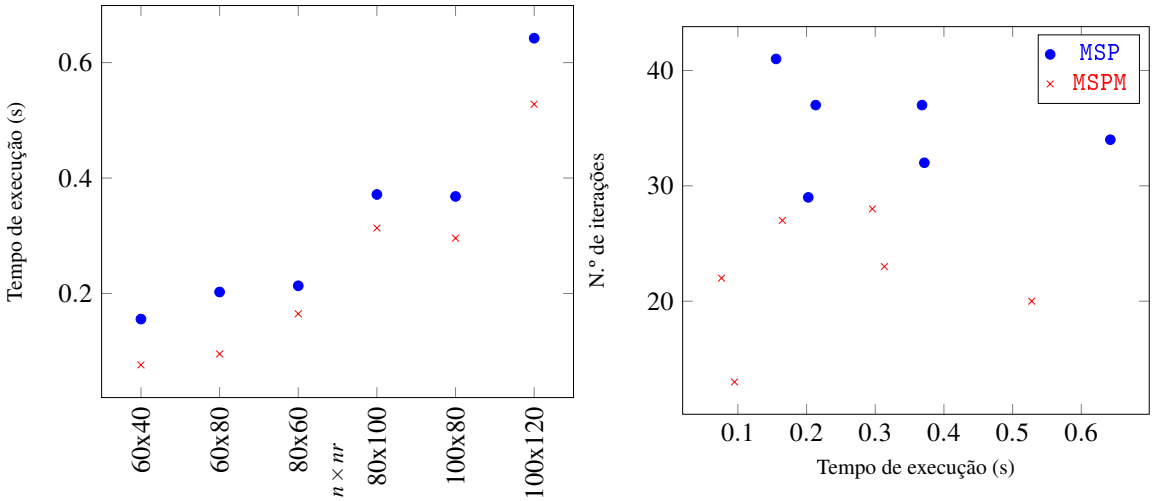
Na Figura 3.5 comparam-se os códigos `MD`, o melhor método paramétrico, com base nos resultados da Secção 2.4, e `MSPM`, o melhor método biobjetivo. Os gráficos registam os quocientes entre o tempo de execução e o número de iterações. Estes valores são todos superiores a 1, o que significa que o código `MD` foi mais eficiente do que o código `MSPM` para todos os casos estudados. Apesar de o método biobjetivo ter sido executado sempre em menos de 0.6 segundos, este demorou até mais de quatro vezes o tempo exigido pelo método paramétrico, e realizou até cinco vezes mais iterações. Em termos do tempo de execução médio, esta melhoria é mais notória para o caso $C = 100$. Em relação ao número de iterações médio, a conclusão depende da dimensão dos problemas, uma vez que quando o número de restrições é superior ao número de variáveis, os valores das iterações efetuadas por cada método são mais próximos quando a variação dos custos é menor, isto é, quando $C = 10$.



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

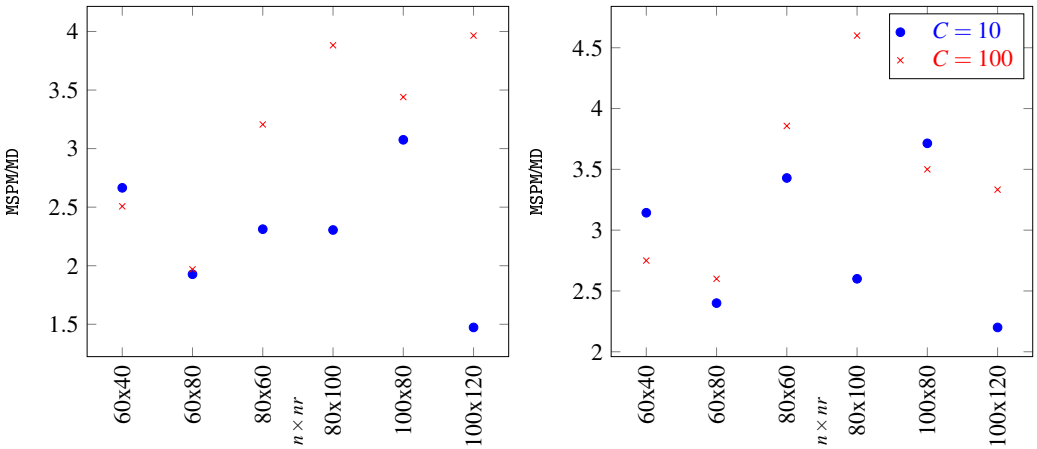
Figura 3.3 Resultados dos métodos `MSP` e `MSPM` para programas lineares fracionários, com $C = 10$



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

Figura 3.4 Resultados dos métodos `MSP` e `MSPM` para programas lineares fracionários, com $C = 100$



(a) Tempo de execução

(b) Número de iterações

Figura 3.5 Resultados comparativos dos métodos `MD` e `MSPM` para programas lineares fracionários

Capítulo 4

Problema fracionário da mochila

Nos dois capítulos anteriores estudaram-se métodos para resolver programas lineares fracionários, primeiro métodos paramétricos iterativos e métodos baseados numa mudança de variáveis, e em seguida métodos baseados numa interpretação dos problemas como problemas biobjetivo. No final de cada capítulo apresentaram-se resultados de testes computacionais de cada um dos dois conjuntos de algoritmos em programas lineares fracionários genéricos. Conforme se referiu anteriormente, os métodos paramétricos, assim como os introduzidos no Capítulo 3, são particularmente úteis, relativamente ao proposto por Charnes e Cooper [5], por se aplicarem igualmente a problemas com estruturas mais específicas, que se perdem com a mudança de variável de Charnes e Cooper.

O presente capítulo tem dois propósitos principais, analisar e comparar o comportamento empírico de métodos paramétricos e métodos biobjetivo, e aplicar os algoritmos a um problema combinatório clássico. O problema escolhido foi o problema da mochila, a sua versão biobjetivo admite soluções eficientes não suportadas. Começa por se descrever resumidamente o problema e algumas suas aplicações. Em seguida, analisam-se os métodos paramétricos e biobjetivo para a resolução do problema da mochila quando a função objetivo é o quociente de funções afins.

4.1 Problema da mochila

A essência do problema da mochila mais simples pode ser enunciada de forma breve. Suponha-se que um viajante pretende levar na sua mochila objetos pelos quais tem diferentes preferências, conhecidas previamente, mas que a mochila tem um limite de peso que consegue suportar. Pretende-se decidir quais os objetos que deve selecionar, por forma a maximizar a soma das suas preferências, mas não excedendo a capacidade da mochila. Esta primeira versão do problema da mochila pode ser formulada do seguinte modo

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s. a} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned} \tag{4.1}$$

onde n é o número de objetos disponíveis, b é a capacidade da mochila e c_j e a_j representam o grau de preferência do viajante pelo objeto j e o peso do mesmo objeto, respetivamente, para $j = 1,\ldots,n$. As variáveis de decisão são binárias, tomando x_j o valor 1 se e somente se o objeto j é incluído na mochila, $j = 1,\ldots,n$. Sem perda de generalidade considera-se $c_j, b > 0$ e $0 \leq a_j \leq b$, para todo o $j = 1,\ldots,n$, e $\sum_{j=1}^{n} a_j > b$.

Apesar da simplicidade desta formulação, o problema da mochila definido em (4.1) é um problema NP-difícil. Este é um dos problemas mais clássicos de otimização linear inteira e tem sido estudado por inúmeros autores, principalmente por se adaptar à modelação de várias situações reais e por surgir como parte da formulação de vários outros problemas. O livro publicado por Martello e Toth [25] sobre este assunto apresenta uma revisão de algumas das versões de problemas da mochila conhecidas atualmente na literatura.

Muitas outras versões do problema da mochila têm sido estudadas. A título de exemplo refere-se o problema da mochila com várias dimensões, definido por

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s. a} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1,\ldots,m \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned}$$

que consiste numa extensão do problema anterior em que os objetos a transportar são obrigados a satisfazer um conjunto de m restrições, todas do mesmo tipo. Neste caso cada objeto j está associado a n parâmetros, a_{ij}, e existe um limite relativamente a cada parâmetro considerado, b_i, para $i = 1,\ldots,m$, $j = 1,\ldots,n$.

Generalizações comuns deste problema consideram variáveis inteiras em vez de binárias, o que permite representar situações em que é possível incluir na mochila vários objetos do mesmo tipo, como na formulação

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s. a} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \geq 0, \quad j = 1,\ldots,n \\ & x_j \in \mathbb{Z}, \quad j = 1,\ldots,n \end{aligned}$$

ou mesmo variáveis com variação real,

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s. a} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \geq 0, \quad j = 1,\ldots,n \end{aligned}$$

Esta última formulação é de grande utilidade, por exemplo, em aplicações à seleção de *portfolios*, caso em que as decisões correspondem ao capital a investir em cada ativo. As restrições do problema

da mochila também podem ser alteradas no sentido de obrigar a mochila a atingir sempre a capacidade máxima, ou seja, por forma a considerar restrições de igualdade em vez de desigualdades.

Destacam-se dois grandes exemplos da aplicação de problemas da mochila, são eles os problemas de corte e os problemas de empacotamento. No primeiro caso o objetivo é definir o corte de grandes objetos para a produção de itens menores em quantidades bem definidas. No segundo procura-se acomodar pequenos objetos em espaços bem definidos. Em ambos os exemplos, a função objetivo do problema define-se medindo as perdas, ou seja, a quantidade de material sobrante, e os vazios, a área ou o volume que ficam por ocupar, respetivamente. Os dois casos podem ser relacionados através de um padrão de corte que minimize a função objetivo [29].

Dos vários processos conhecidos para resolver o problema da mochila com variáveis binárias destacam-se métodos de enumeração implícita com *branch-and-bound* [14], métodos de programação dinâmica, baseados na interpretação do problema como um problema do caminho mais curto num grafo construído de acordo com as decisões permitidas para a seleção de objetos [2] e métodos híbridos [28].

4.2 Problema fracionário da mochila

Os problemas da mochila referidos na secção anterior lidam com uma função objetivo que é linear. Contudo, podem também ser tratados no sentido de otimizar um quociente de funções afins, com a seguinte formulação

$$\max \quad \frac{\sum_{j=1}^{n} N_j x_j + \alpha}{\sum_{j=1}^{n} D_j x_j + \beta} \tag{4.2a}$$

$$\text{s. a} \quad \sum_{j=1}^{n} a_j x_j \leq b \tag{4.2b}$$

$$x_j \in \{0,1\}, \quad j = 1,\ldots,n \tag{4.2c}$$

onde $N, D, a \in \mathbb{R}^n$, com $D > 0$, $\alpha, \beta > 0$ e $b \in \mathbb{R}$. Este problema pode ser resolvido pelos métodos paramétricos estudados no Capítulo 2 e pode igualmente ser resolvido pelos métodos de soma ponderada apresentados no Capítulo 3, associados ao problema linear biobjetivo da mochila

$$\begin{aligned} \max \quad & N(x) = \sum_{j=1}^{n} N_j x_j \\ \min \quad & D(x) = \sum_{j=1}^{n} D_j x_j \\ \text{s. a} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned} \tag{4.3}$$

Numa segunda fase considera-se a relaxação linear do problema (4.2), isto é, o problema fracionário da mochila que se obtém substituindo as condições (4.2c) por

$$0 \leq x_j \leq 1, \quad j = 1, \ldots, n.$$

Alguns trabalhos dedicaram-se ao estudo do problema fracionário da mochila. Destes, destacam-se [19], em que Ishii et al. estudam algumas propriedades do problema na sua versão inteira, e [17], em que Ibaraki testa o comportamento de novos métodos paramétricos considerando o problema na sua versão real.

4.3 Experiência computacional

Para comparar os métodos descritos nos Capítulos 2 e 3 em termos de eficiência, apresentam-se em seguida resultados de alguns testes computacionais. Pretende-se comparar os vários métodos para um problema concreto, tendo-se optado pelo problema da mochila. Fica fora do âmbito deste trabalho o estudo de um método dedicado exclusivamente a este problema. Por este motivo recorreu-se ao *solver* CPLEX 12.8 para resolver as formulações de problemas da mochila apresentadas na secção anterior. Selecionaram-me os métodos que mostraram melhores resultados para programas lineares fracionários, apresentados nas Secções 2.4 e 3.4, e adaptaram-se os códigos `MD`, `MPB` e `MPBM`, no primeiro caso, e `MSP` e `MSPM`, no segundo. Os métodos foram aplicados a dois conjuntos de problemas da mochila fracionários:

Testes A problemas da mochila fracionários com variáveis binárias, como o definido em (4.2),

Testes B a relaxação linear dos problemas no ponto anterior, com variáveis reais satisfazendo as restrições $0 \leq x_j \leq 1$, para $j = 1, \ldots, n$.

No primeiro conjunto de testes os subproblemas da mochila binários foram resolvidos através da rotina `cplexbilp`, enquanto que no segundo se aplicou a rotina `cplexlp`. Os testes foram executados na máquinha referida anteriormente.

Os problemas gerados dependem dos seguintes parâmetros:

1. $N = [N_j]_{j=1,\ldots,n}$, $D = [D_j]_{j=1,\ldots,n}$, $a = [a_j]_{j=1,\ldots,n}$, com N_j, D_j e a_j gerados aleatoriamente em $\{1, \ldots, C\}$, $j = 1 \ldots, n$;
2. $\alpha = 0$ e β gerado de forma uniforme em $\{1, \ldots, C\}$;
3. b escolhido aleatoriamente em $\{1, \ldots, \sum_{j=1}^{n} a_j\}$.

Além disso, utilizaram-se os limites $C = 10, 100$.

Testes A As Tabelas 4.1 e 4.2 resumem os resultados médios obtidos pelos cinco códigos para 30 problemas fracionários da mochila de cada dimensão.

Considerando $C = 10$, o método com melhores resultados foi o `MD`, alcançando o menor tempo de execução médio e o mais baixo número de iterações, para quase todos os casos. Estes resultados são ainda melhores quando $C = 100$, caso em que `MD` foi sempre, em média, mais rápido do que os

Tabela 4.1 Tempo de execução (em segundos) e número de iterações em problemas fracionários binários da mochila, com $C = 10$

	MD			MPB			MPBM			MSP			MSPM		
n	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
60	0.1420	0.0696	6	0.5568	1.2728	53	0.7939	2.0138	37	0.9495	0.1767	52	0.1628	0.0449	9
80	0.1380	0.0526	6	0.4040	0.8789	37	1.6127	5.0839	54	1.2363	0.2202	63	0.1784	0.0541	10
100	0.2063	0.0793	6	0.4193	0.8942	37	0.2269	0.1586	4	1.3362	0.3051	66	0.1804	0.0415	10
120	0.1614	0.0798	6	0.7260	1.5259	69	0.2155	0.0686	4	1.6083	0.2461	82	0.1977	0.0722	10
140	0.1686	0.0949	6	1.0344	2.3034	68	0.5547	1.4083	21	1.6832	0.4412	77	0.1951	0.0484	10
160	0.1539	0.0601	6	0.7942	2.1306	52	0.8146	1.7380	37	1.7645	0.4296	81	0.2170	0.0514	10

Tabela 4.2 Tempo de execução (em segundos) e número de iterações em problemas fracionários binários da mochila, com $C = 100$

	MD			MPB			MPBM			MSP			MSPM		
n	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
60	0.1069	0.0435	6	0.2699	0.0842	24	0.2162	0.0759	4	1.4077	0.4308	84	0.1670	0.0509	10
80	0.1171	0.0431	6	0.2737	0.0640	25	0.2463	0.0959	4	1.6753	0.3342	109	0.1836	0.0729	11
100	0.1329	0.0401	6	0.2882	0.1170	24	0.2894	0.0985	4	2.1298	0.4855	123	0.1973	0.0493	11
120	0.1130	0.0596	7	0.4215	0.8747	40	0.2844	0.1629	4	3.1839	0.6645	175	0.1845	0.0524	11
140	0.1167	0.0425	6	0.2870	0.1072	24	0.5658	1.0139	21	2.9845	0.7841	170	0.2118	0.0501	11
160	0.1204	0.0425	7	0.2910	0.0855	24	2.1751	8.5646	38	4.2003	1.0838	205	0.2126	0.0471	12

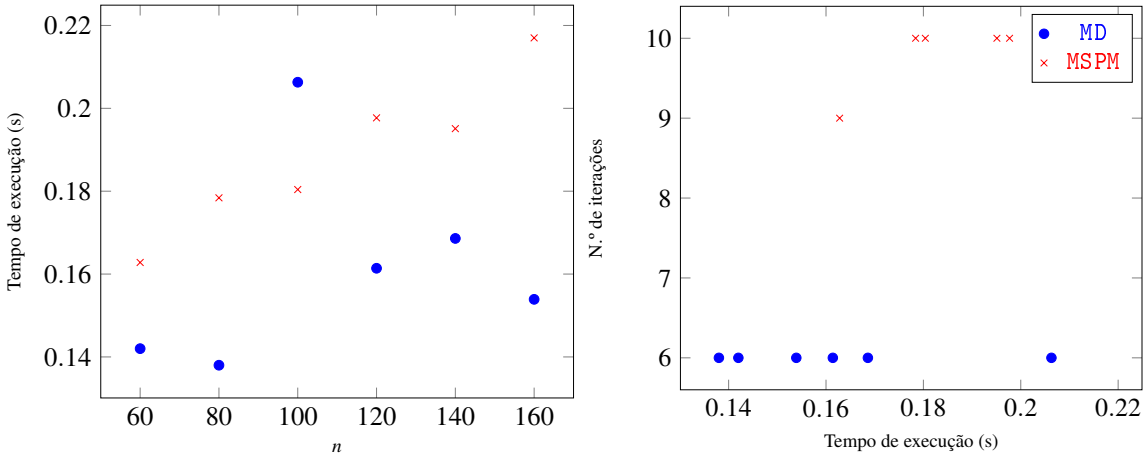
restantes códigos. Além disso, apesar de ter sido mais lento, o método `MPBM` destacou-se em relação ao número de iterações nos problemas de dimensões menores, enquanto que para as mais elevadas o método `MD` foi o que necessitou de menos iterações para encontrar a solução ótima.

No que respeita aos métodos biobjetivo, voltou a confirmar-se uma diminuição significativa do número de iterações, e também do tempo de execução ao utilizar o código `MSPM`, relativamente a `MSP`, principalmente para os problemas com maior variação de custos, $C = 100$. A situação foi análoga para os códigos `MPBM` e `MPB`, ainda que de forma menos coerente. É evidente que de entre os métodos biobjetivo, o `MSPM` é o que se aproxima mais da eficiência conseguida pelo `MD`. Em praticamente todos os casos, os métodos paramétricos mostram ser mais eficientes, comparativamente com o `MSP`.

Nas Figuras 4.1a e 4.2a observa-se um comportamento praticamente linear do tempo de execução médio do `MSPM` relativamente à dimensão dos problemas. No caso do `MD`, o comportamento é próximo de uma constante para $C = 100$.

Testes B As Tabelas 4.3 e 4.4 resumem os resultados dos mesmos métodos para o segundo conjunto de testes, relativamente à versão relaxada do problema fracionário da mochila com variáveis binárias.

De um modo geral estes problemas foram resolvidos mais facilmente do que aqueles que têm variáveis binárias, tendo exigido menos iterações do que os anteriores (exceto no caso do método `MPB`). Após a análise dos resultados, verifica-se que a ordem de grandeza dos tempos de execução e número de iterações não se altera grandemente quando se alarga o intervalo de custos permitidos. Além disso, ao contrário do que foi observado para programas lineares fracionários genéricos – Secção 2.4 – o código `MPBM` foi o mais eficiente na resolução dos problemas da mochila quando $C = 10$, tanto em termos de tempo de execução como de número de iterações. Em relação a $C = 100$, observa-se que existe uma divergência nos resultados, chegando o `MSP` a ser o código mais eficiente para algumas



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

Figura 4.1 Resultados dos métodos `MD` e `MSPM` para problemas da mochila fracionários com variáveis binárias, com $C = 10$

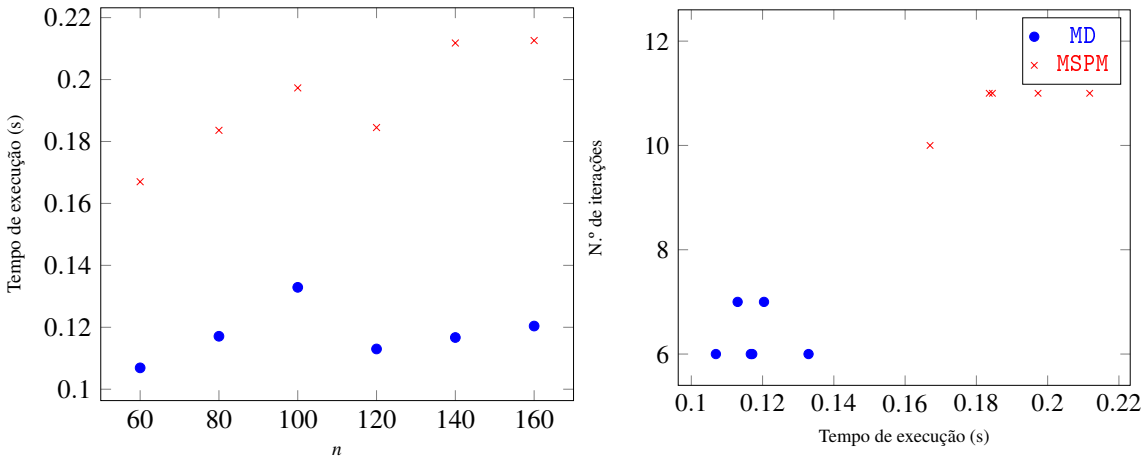
Tabela 4.3 Tempo de execução (em segundos) e número de iterações em problemas fracionários da mochila, com $C = 10$

	MD			MPB			MPBM			MSP			MSPM		
n	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
60	0.0526	0.1919	6	0.0698	0.1544	55	0.0113	0.0060	4	0.0135	0.0328	5	0.0134	0.0328	5
80	0.0295	0.0932	6	0.0459	0.0868	39	0.0103	0.0037	4	0.0148	0.0358	5	0.0148	0.0367	5
100	0.0138	0.0357	6	0.0473	0.0905	39	0.0149	0.0094	4	0.0142	0.0340	5	0.0148	0.0346	5
120	0.0150	0.0352	6	0.0961	0.1920	71	0.0139	0.0061	4	0.0140	0.0322	5	0.0152	0.0338	5
140	0.0182	0.0510	6	0.0879	0.1655	71	0.0122	0.0047	4	0.0169	0.0320	5	0.0148	0.0336	5
160	0.0137	0.0341	6	0.0770	0.1746	55	0.0128	0.0042	4	0.0153	0.0341	5	0.0156	0.0358	5

dimensões, o que contradiz os resultados anteriores. Nestes últimos problemas, alguns métodos exigiram mais iterações do que para $C = 10$. Apesar disso, o código `MD` continuou a ser, quase sempre, o mais rápido para os casos com dimensões maiores.

Para esta classe de problemas há uma maior homogeneidade de resultados, em termos gerais. Em particular, os métodos biobjetivo foram praticamente tão eficientes quanto os métodos paramétricos testados. É ainda de salientar que, à exceção do código `MPB`, todos os restantes códigos necessitaram de menos iterações para atingir a solução ótima quando as variáveis são reais, ainda que a diferença seja pouco marcada.

Apesar de não ser tão notório como no resultados computacionais do código `MPBM` apresentados no Capítulo 2.4, observou-se que em alguns problemas o código `MPB` não foi capaz de identificar de imediato ter encontrado uma aproximação razoável da solução ótima, tendo gerado mais iterações do que as necessárias. Isto aconteceu, por exemplo, para os problemas registados na Tabela 4.4, tendo sido o método interrompido ao fim de 26 iterações, apesar de ter atingido a solução ótima nas primeiras.



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

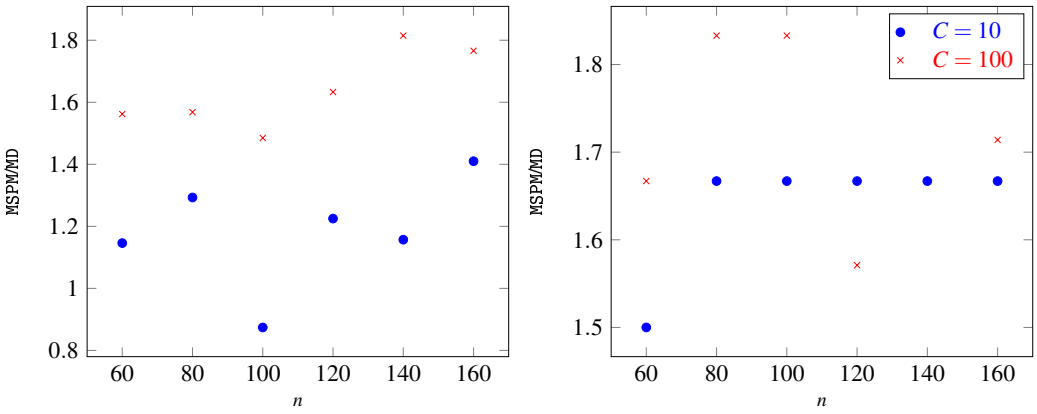
Figura 4.2 Resultados dos métodos `MD` e `MSPM` para problemas da mochila fracionários com variáveis binárias, com $C = 100$

Tabela 4.4 Tempo de execução (em segundos) e número de iterações em problemas fracionários da mochila, com $C = 100$

	MD			MPB			MPBM			MSP			MSPM		
n	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$	$\bar{t}$	s_t	$\bar{k}$
60	0.0262	0.1027	6	0.0438	0.0832	26	0.0104	0.0043	5	0.0142	0.0369	5	0.0140	0.0322	5
80	0.0149	0.0431	6	0.0321	0.0349	26	0.0120	0.0062	5	0.0148	0.0377	5	0.0139	0.0337	5
100	0.0149	0.0392	6	0.0336	0.0355	26	0.0215	0.0568	8	0.0140	0.0320	5	0.0144	0.0347	5
120	0.0147	0.0352	7	0.0560	0.0947	42	0.0130	0.0052	5	0.0143	0.0325	5	0.0143	0.0331	5
140	0.0145	0.0343	6	0.0352	0.0376	26	0.0146	0.0062	5	0.0148	0.0328	5	0.0171	0.0353	5
160	0.0149	0.0359	7	0.0339	0.0348	26	0.0150	0.0058	5	0.0149	0.0364	5	0.0153	0.0324	5

As Figuras 4.4a e 4.5a mostram os mesmos resultados, onde se destaca um comportamento quase constante dos tempos de execução do código `MPBM`, quando $C = 10$, e um aparente decréscimo do código `MD` à medida que aumenta a dimensão do problema, em ambos os casos.

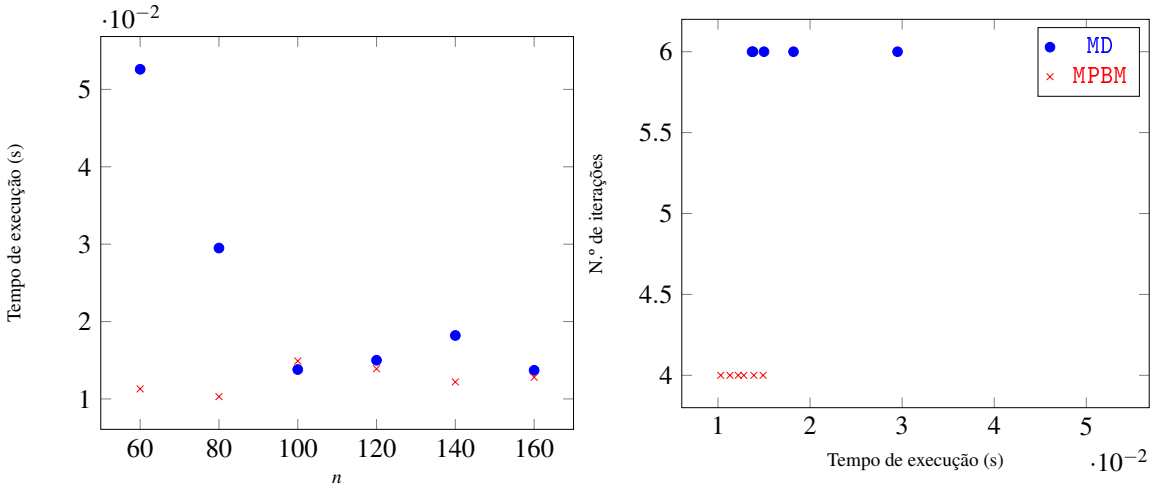
Nas Figuras 4.3 e 4.8 comparam-se os códigos `MD` e `MSPM`, por terem sido os mais eficientes e consistentes das duas classes, paramétrica e biobjetivo. Tanto para os problemas binários como para os problemas relaxados, os resultados dos doi códigos foram próximos, sendo cada um deles o mais rápido para diferentes dimensões. Por exemplo, o código `MD` foi o melhor para a maior parte dos problemas da mochila binários, enquanto que o código `MSPM` encontrou a solução em menos iterações e menos tempo de execução do que `MD` na maioria dos problemas da mochila relaxados. Observa-se contudo que, ao contrário do código `MD`, o código `MSPM` é mais sensível ao aumento do valor de C, o que pode ser consequência do aumento do número de soluções eficientes calculadas.



(a)
Tempo de execução

(b)
Número de iterações

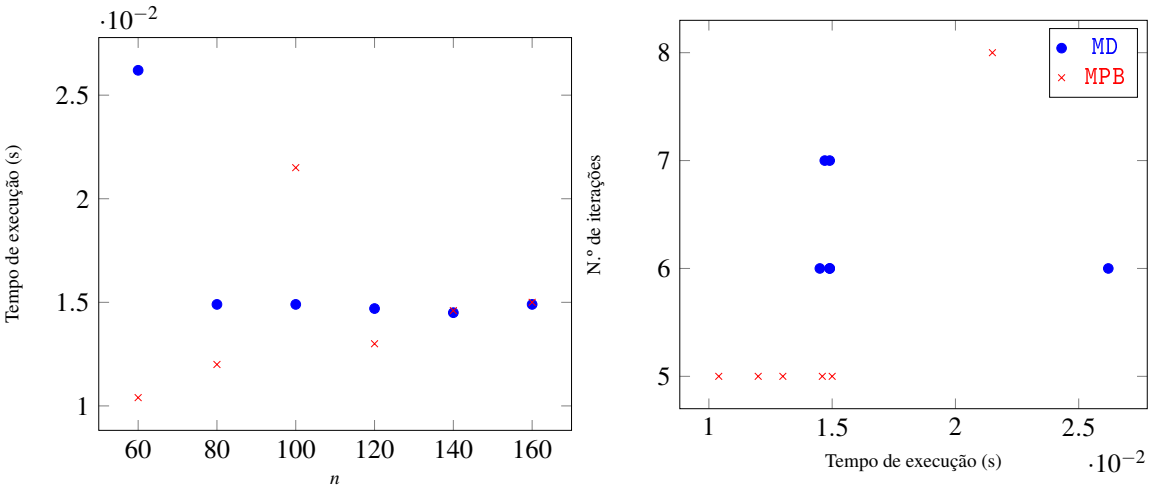
Figura 4.3 Resultados comparativos dos métodos `MD` e `MSPM` para programas da mochila fracionários com variáveis binárias



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

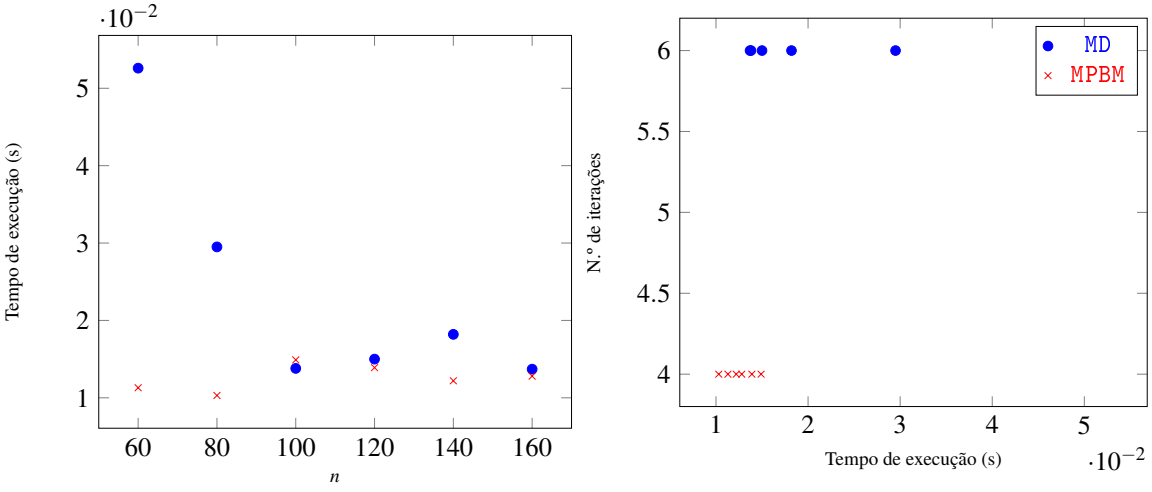
Figura 4.4 Resultados dos métodos `MD` e `MPBM` para problemas da mochila fracionários com variáveis reais, com $C = 10$



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

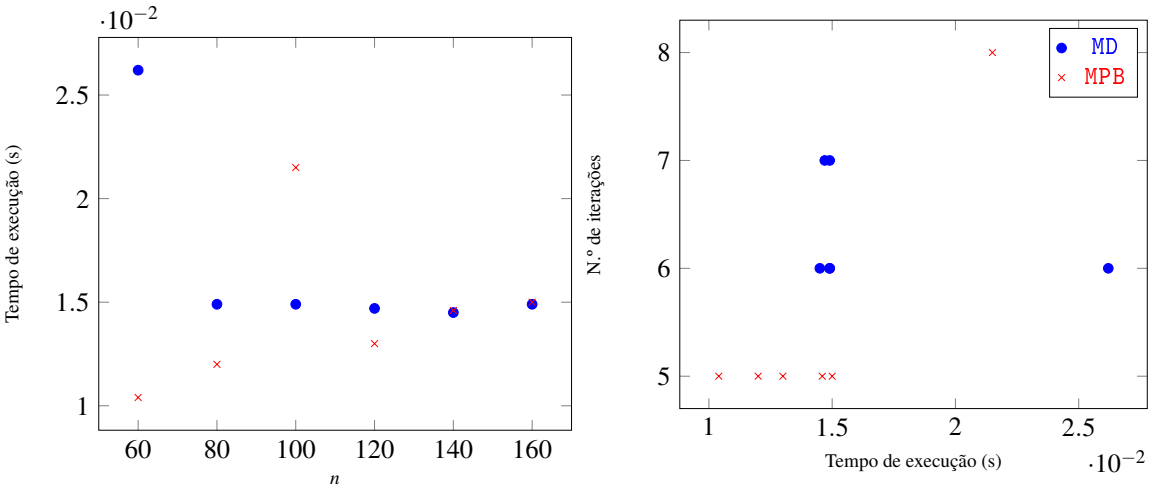
Figura 4.5 Resultados dos métodos `MD` e `MPBM` para problemas da mochila fracionários com variáveis reais, com $C = 100$



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

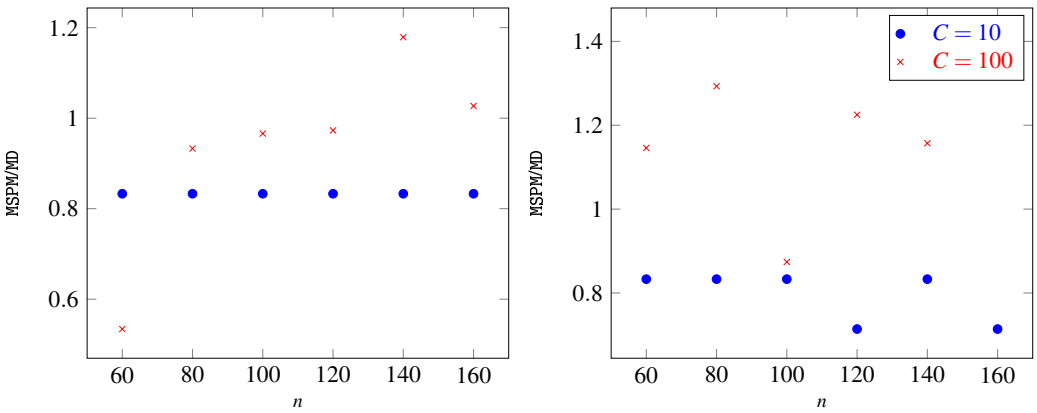
Figura 4.6 Resultados dos métodos `MD` e `MPBM` para problemas da mochila fracionários com variáveis reais, com $C = 10$



(a)
Tempo *versus* dimensão

(b)
Tempo *versus* número de iterações

Figura 4.7 Resultados dos métodos `MD` e `MPBM` para problemas da mochila fracionários com variáveis reais, com $C = 100$



(a)
Tempo de execução

(b)
Número de iterações

Figura 4.8 Resultados comparativos dos métodos `MD` e `MSPM` para programas da mochila fracionários com variáveis reais

Capítulo 5

Conclusão

Os problemas lineares fracionários caracterizam-se por terem um quociente de funções afins como função objetivo e um conjunto de restrições lineares. No início do presente trabalho reviram-se algumas propriedades deste tipo de problemas e os principais métodos conhecidos para o resolver: métodos paramétricos e métodos de mudança de variável. Testes computacionais destes métodos num conjunto de problemas gerados aleatoriamente destacaram os métodos de Charnes e Cooper como sendo o mais rápido e o de Dinkelbach como o método paramétrico mais eficiente.

Em seguida, estudaram-se alguns conceitos de otimização biobjetivo e a sua relação com a programação linear fracionária, do que resultou a proposta de resolução destes problemas baseada no cálculo de soluções eficientes suportadas do problema biobjetivo associado. Este método depende da resolução de problemas semelhantes ao inicial, mas com somas ponderadas das funções do quociente como função objetivo. Uma versão modificada do método da soma ponderada foi também apresentada. Esta modificação tem por base propriedades dos problemas lineares fracionários revistas no Capítulo 2 e permite reduzir, de forma significativa, o número de soluções calculadas e o número de iterações do método. Estes resultados foram verificados empiricamente através de experiência computacional para vários exemplos de programas lineares fracionários com diferentes dimensões.

De um modo geral, os problemas testados foram resolvidos em menos de dois segundos. Apesar de o método da soma ponderada modificado, `MSPM`, se ter revelado consideravelmente mais eficiente do que a sua primeira versão, uma comparação entre os resultados reportados nos Capítulos 2 e 3 indica que para programas lineares fracionários genéricos os métodos biobjetivo, nomeadamente `MSPM`, exigiram mais iterações e maior tempo de execução do que o melhor método paramétrico para os mesmos testes, o método de Dinkelbach, `MD`. Em termos de tempo de CPU o código `MD` foi cerca de duas a quatro vezes mais rápido do que `MSPM`.

Uma das principais vantagens dos métodos paramétricos e biobjetivo relativamente à transformação do problema num programa linear é o facto de não alterarem a região admisível e, portanto, determinarem soluções do tipo original. Por este motivo, o Capítulo 4 é dedicado à comparação entre os métodos paramétricos e os métodos biobjetivo, para um problema combinatório clássico: o problema da mochila. Considerou-se a sua versão fracionária com variáveis binárias e com variáveis reais. As conclusões não coincidiram completamente com as anteriores. Neste caso os melhores resultados foram obtidos pelos métodos `MD`, `MPBM` e `MSPM`. Relativamente aos métodos `MPB` e `MPBM` detectaram-se casos em que a solução ótima é atingida, mas não é identificada assim que é encontrada,

obrigando à execução de iterações sem que se altere a solução. Isto ocorreu para problemas genéricos e para o problema da mochila, respetivamente.

Só pontualmente o método biobjetivo `MSPM` foi o mais eficiente de todos, além de se ter mostrado mais sensível à variação dos valores de custo dos problemas do que os métodos paramétricos. Contudo, de um modo geral os resultados não foram muito diferentes dos restantes algoritmos e foi largamente mais competitivo do que o método `MD` para problemas da mochila com variáveis reais, o que indica que pode ser uma alternativa de resolução de programas lineares fracionários a considerar.

Como trabalho futuro pretende-se analisar diferentes critérios de paragem dos algoritmos e a sua influência nos resultados apresentados por cada um. É igualmente relevante alargar a experiência computacional a problemas com dimensão maior do que os já apresentados e analisar o comportamento de alguns dos algoritmos considerados para outras classes de problemas combinatórios.

Bibliografia

[1] Ahuja, R. (1988). Minimum cost-reliability ratio path problem. *Computers & Operations Research*, 15(1):83–89.

[2] Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, NJ.

[3] Bereanu, B. (1964). Programme de risque minimal en programmation linéaire stochastique. In *Comptes rendus de l'academie des sciences*, volume 259, pages 981–983. Serie I, Mathematique.

[4] Boussofiane, A., Dyson, R. G., and Thanassoulis, E. (1991). Applied data envelopment analysis. *European Journal of Operational Research*, 52(1):1–15.

[5] Charnes, A. and Cooper, W. W. (1962). Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186.

[6] Charnes, A., Cooper, W. W., and Rhodes, E. (1981). Evaluating program and managerial efficiency: an application of data envelopment analysis to program follow through. *Management Science*, 27(6):668–697.

[7] Clímaco, J., Antunes, C. H., and Alves, M. J. (2003). *Programação linear multiobjectivo: do modelo de programação linear clássico à consideração explícita de várias funções objectivo*. Coimbra University Press.

[8] Cohon, J. (1978). *Multiobjective programming and planning*. Academic Press, New York.

[9] Cooper, W. W., Seiford, L. M., and Zhu, J. (2004). Data envelopment analysis. In *Handbook on data envelopment analysis*, pages 1–39. Springer.

[10] Derman, C. (1962). On sequential decisions and markov chains. *Management Science*, 9(1):16–24.

[11] Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science*, 13(7):492–498.

[12] Ehrgott, M. (2005). Multicriteria optimization. *Management Science*, pages 1–8.

[13] Fox, B. L. (1969). Finding minimum time-cost ratio circuits. *Operations Research*, 17:546–551.

[14] Gilmore, P. C. and Gomory, R. E. (1963). A linear programming approach to the cutting stock problem – part ii. *Operations research*, 11(6):863–888.

[15] Guerriero, F. and Pugliese, L. D. P. (2011). Multi-dimensional labelling approaches to solve the linear fractional elementary shortest path problem with time windows. *Optimization Methods & Software*, 26(2):295–340.

[16] Gupta, N. (2019). Optimization of fuzzy bi-objective fractional assignment problem. *OPSEARCH*, pages 1–12.

[17] Ibaraki, T. (1983). Parametric approaches to fractional programs. *Mathematical Programming*, 26:345–362.

[18] Isbell, J. R. and Marlow, W. H. (1956). Attrition games. *Naval Research Logistics Quarterly*, 3(1-2):71–94.

[19] Ishii, H., Ibaraki, T., and Mine, H. (1977). Fractional knapsack problems. *Mathematical Programming*, 13:255–271.

[20] Kabadi, S. N. and Punnen, A. P. (2008). A strongly polynomial simplex method for the linear fractional assignment problem. *Operations Research Letters*, 36(4):402–407.

[21] Klein, M. (1962). Inspection—maintenance—replacement schedules under markovian deterioration. *Management Science*, 9(1):25–32.

[22] Lasdon, L. S. (2002). *Optimization theory for large systems*. Courier Corporation.

[23] Lin, C.-J. (2015). A simplex-based labelling algorithm for the linear fractional assignment problem. *Optimization*, 64(4):929–939.

[24] Lin, C.-J. and Wen, U.-P. (2004). A labeling algorithm for the fuzzy assignment problem. *Fuzzy Sets and Systems*, 142(3):373–391.

[25] Martello, S. and Toth, P. (1990). *Knapsack problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester.

[26] Mjelde, K. M. (1978). Allocation of resources according to a fractional objective. *European Journal of Operational Research*, 2(2):116–124.

[27] Mjelde, K. M. (1983). *Methods of the allocation of limited resources*. John Wiley & Sons.

[28] Plateau, G. and Elkihel, M. (1985). A hybrid algorithm for the 0-1 knapsack problem. *Methods of Operations Research*, 49:277–293.

[29] Prado Marques, F. and Nereu Arenales, M. (2002). O problema da mochila compartimentada e aplicações. *Pesquisa Operacional*, 22(3):285–304.

[30] Radzik, T. (2013). *Fractional Combinatorial Optimization*, pages 1311–1355. Springer New York, New York, NY.

[31] Roan, J. and Lee, C. (2003). Algorithms for linear fractional shortest path problem with time windows. *Pan-Pacific Management Review*, 6(1):75–84.

[32] Schaible, S. and Ibaraki, T. (1983). Fractional programming. *European Journal of Operation Research*, 12:325–338.

[33] Shen, K. and Yu, W. (2018a). Fractional programming for communication systems—part I: Power control and beamforming. *IEEE Transactions on Signal Processing*, 66(10):2616–2630.

[34] Shen, K. and Yu, W. (2018b). Fractional programming for communication systems—part II: Uplink scheduling via matching. *IEEE Transactions on Signal Processing*, 66(10):2631–2644.

[35] Shigeno, M., Saruwatari, Y., and Matsui, T. (1995). An algorithm for fractional assignment problems. *Discrete Applied Mathematics*, 56(2-3):333–343.

[36] Soroush, H. (2008). Optimal paths in bi-attribute networks with fractional cost functions. *European Journal of Operational Research*, 190(3):633–658.

[37] Stancu-Minasian, I. M. (2017). A eighth bibliography of fractional programming. *Optimization*, 66(3):439–470.

[38] Swarup, K. (1966). Transportation technique in linear fractional functional programming. *Journal of royal naval scientific service*, 21(5):256–260.

[39] Ziemba, W. T., Parkan, C., and Brooks-Hill, R. (2013). Calculation of investment portfolios with risk free borrowing and lending. In *Handbook of the Fundamentals of Financial Decision Making: Part I*, pages 375–388. World Scientific.