PROGRAMAÇÃO LINEAR INTEIRA E INTEIRA-MISTA MULTIOBJETIVO: CONCEITOS FUNDAMENTAIS E MÉTODOS

Maria João Alves
Fac. de Economia da Universidade de Coimbra / INESC - Coimbra
Av. Dias da Silva, nº 165, 3004-512 Coimbra, PORTUGAL
mjalves@fe.uc.pt

João Paulo Costa
Fac. de Economia da Universidade de Coimbra / INESC - Coimbra
Av. Dias da Silva, nº 165, 3004-512 Coimbra, PORTUGAL
jpaulo@fe.uc.pt

RESUMO

Neste artigo apresentam-se os conceitos fundamentais de programação linear inteira e inteira-mista multiobjetivo, faz-se uma breve revisão de métodos geradores de soluções eficientes, para este tipo de problemas, e mostram-se alguns novos desenvolvimentos na área. De entre os desenvolvimentos mais recentes, destacamos a exploração do espaço dos pesos na pesquisa de soluções eficientes suportadas, desenvolvida por nós ou por outros autores.

PALAVARAS CHAVE. Otimização multiobjetivo. Programação inteira mista multiobjetivo. Métodos geradores.

ABSTRACT

This paper presents the fundamental concepts in multiobjective linear integer and mixed-integer programming, reviews generating methods of efficient solutions for this type of problems and shows some new developments in the area. Among the most recent developments, we highlight procedures that exploit the weight space for computing supported efficient solutions, which have been developed by us or by other authors.

KEYWORDS. Multiobjetive optimization. Multiobjective mixed integer programming. Generating methods.

1. Introdução

O problema de programação linear inteira multiobjetivo (PLIMO), em que todas as variáveis de decisão são inteiras, ou inteira-mista multiobjetivo (PLIMMO), em que só algumas das variáveis são inteiras, pode definir-se da seguinte forma:

$$\left.\begin{array}{l} \max z_1 = f_1(x) = c^1 x \\ \quad\quad ... \\ \max z_p = f_p(x) = c^p x \end{array}\right\} \text{"Max" } z = \mathrm{f}(x) = Cx \qquad (1)$$

$$\text{s.a:} \quad x \in X = \left\{ x \in \Re^n \mid Ax = b, x \geq 0, x_j \text{ inteira}, j \in I \right\}$$

I é o conjunto dos índices das variáveis inteiras, $I \subseteq \{1,...,n\}$, $I \neq \varnothing$. C é a matriz dos coeficientes das funções objetivo cujas linhas são os vetores $(1\times n)$ c^k, $k=1,...,p$. A é a matriz $(m\times n)$ dos coeficientes técnicos das restrições e b é o vetor $(m\times 1)$ dos termos independentes das restrições. Assume-se que a região admissível X é não-vazia e limitada.

A existência de variáveis inteiras em modelos de programação multiobjetivo acrescenta vários tipos de dificuldades na resolução dos problemas, mesmo em problemas lineares. Por exemplo, a região admissível deixa de ser convexa e, por isso, o desenvolvimento de métodos para problemas PLIMO/PLIMMO está para além de uma simples combinação de métodos de programação linear multiobjetivo com técnicas de cálculo de soluções inteiras.

Neste artigo, começaremos por expor, na secção 2, os conceitos fundamentais de programação multiobjectivo, em particular, para os casos PLIMO e PLIMMO, e apresentaremos processos de cálculo de soluções eficientes usando funções escalares substitutas (técnicas de escalarização). Na secção 3 discutiremos métodos para PLIMO/PLIMMO com particular destaque para métodos *geradores*, que têm como intenção gerar todas as soluções eficientes do problema ou um subconjunto pré-definido (como, por exemplo, as soluções eficientes suportadas). Abordaremos apenas métodos exatos que se destinam a problemas genéricos, não incluindo métodos heurísticos ou meta-heurísticos, ou métodos especializados para problemas com uma estrutura particular (como, por exemplo, problemas de fluxo em redes). Por fim, na secção 4, apresentaremos desenvolvimentos recentes na área baseados na exploração do espaço dos pesos, incluindo algum trabalho realizado por nós.

2. Conceitos fundamentais

Tal como em outros problemas de programação multiobjectivo, em PLIMO/PLIMMO não existe, em geral, uma solução admissível $x \in X$ que otimize simultaneamente todas as funções objetivo. O conceito de eficiência é, assim, definido da forma habitual:

Uma solução $x' \in X$ é *eficiente* sse não existe uma outra solução $x \in X$ tal que $f_k(x) \geq f_k(x')$ para todo o $k=1,...,p$ e $f_k(x) > f_k(x')$ para pelo menos um k. Seja X_E o conjunto das soluções eficientes.

Uma solução $x' \in X$ é *fracamente eficiente* se e só se não existir uma outra solução $x \in X$ tal que $f_k(x) > f_k(x')$ para todo o $k=1,...,p$. Por definição, o conjunto das soluções fracamente eficientes inclui o conjunto das soluções eficientes, definido antes. No entanto, por razões de simplicidade de linguagem, quando nos referimos a soluções fracamente eficientes não estamos a incluir as soluções eficientes.

Seja $Z \subset \Re^p$ a imagem da região admissível X no espaço das funções objetivo, tal que cada ponto $x \in X$ é mapeado no ponto $z = \mathrm{f}(x)$. Um ponto $z' \in Z$ correspondente a uma solução x' eficiente diz-se *não dominado* (ou fracamente não dominado se x' for fracamente eficiente).

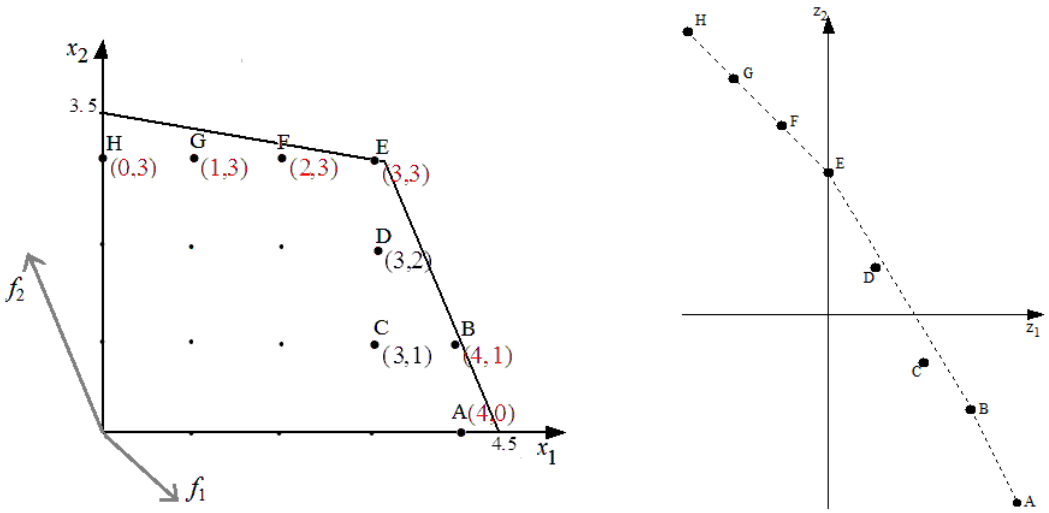
O conjunto de todos o pontos não dominados é $Z_{ND} = \{z' = \mathrm{f}(x') \in Z: \ x' \in X_E\}$.

Um conceito importante em PLIMO/PLIMMO é a distinção entre soluções eficientes *suportadas* e *não suportadas*. Um ponto não dominado $z' \in Z_{ND}$ é *não suportado* se for

dominado por alguma combinação convexa (não admissível) de pontos pertencentes a Z_{ND}. Todos os pontos não dominados não suportados estão localizados na fronteira do invólucro convexo de Z (*conv Z*). A um ponto $z'=f(x')$ não dominado não suportado corresponde uma solução x' *eficiente não suportada*. As outras soluções eficientes dizem-se *suportadas*. Podemos ainda distinguir duas classes de soluções *eficientes/não dominadas suportadas*: soluções cujos pontos não dominados $z \in Z$ são vértices de *conv Z*, a que chamamos soluções *extremas*, e soluções cujos pontos não dominados $z \in Z$ estão no interior relativo de uma face de *conv Z*, a que chamamos *não extremas*.

Para ilustrar estes conceitos, consideremos o seguinte exemplo de PLIMO (e*xemplo 1*): {max $z_1 = x_1 - x_2$; max $z_2 = -x_1 + 2x_2$: $x_1 + 6x_2 \leq 21$; $14x_1 + 6x_2 \leq 63$; $x_1, x_2 \geq 0$ e inteiras}.

A figura 1 mostra as soluções eficientes deste problema, suportadas e não suportadas, no espaço das variáveis de decisão (a) e no espaço dos objetivos (b). A fronteira do invólucro convexo de Z (*conv Z*), sobre a qual estão os pontos não dominados suportados, está assinalada a tracejado na figura 1(b). As soluções *A*, *B*, *E*, *F*, *G* e *H* são soluções eficientes *suportadas*, enquanto que *C* e *D* são eficientes *não suportadas*. Como se pode observar na figura 1(b), estas soluções são dominadas por algumas combinações convexas – não admissíveis – de *B* e *E*. De entre o conjunto das soluções eficientes suportadas, podemos ainda distinguir as soluções extremas *A*, *B*, *E* , *H* (vértices de *conv Z*) e as soluções não extremas *F* e *G* que estão sobre o segmento de reta de *conv Z* que une *E* a H.



(a) Espaço das variáveis de decisão (b) Espaço dos objetivos

Fig. 1 – Soluções eficientes suportadas e não suportadas do e*xemplo* 1.

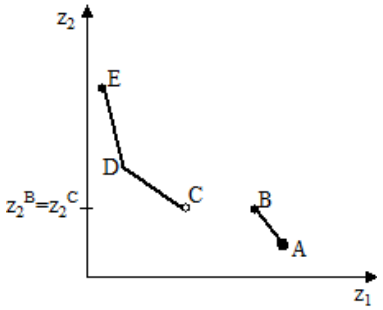


Fig. 2 – Ilustração de diferentes tipos de soluções eficientes em PLIMMO.

A figura 2 ilustra estes conceitos para um problema de PLIMMO (com duas funções objetivo a maximizar), mostrando a fronteira não dominada no espaço dos objetivos. Os pontos

A, *B* e *E* (extremos) e os pontos (não extremos) do segmento *AB* são não dominados *suportados*; os segmentos *CD* e *DE*, excluindo os pontos *C* e *E*, são não dominados *não suportadas* porque são dominados por combinações convexas (não admissíveis) de *B* e *E*; *C* é fracamente não dominado porque não existe nenhum outro ponto que seja estritamente melhor do que ele nas duas funções objetivo, mas é dominado por *B* que tem valor igual em z_2 e valor superior em z_1.

Ao contrário do que acontece em programação linear multiobjetivo (PLMO), não é possível caraterizar por completo o conjunto das soluções eficientes de problemas PLIMO/PLIMMO usando somas pesadas das funções objetivo, porque as soluções eficientes não suportadas não são alcançáveis por este processo de escalarização. Já no que diz respeito às soluções eficientes suportadas, para cada uma existe sempre um vetor de pesos $\lambda \in \Lambda = \{ \lambda \in \Re^p : \lambda_k > 0,\ k=1,...,p,\ \sum_{k=1}^{p} \lambda_k = 1 \}$ tal que a solução eficiente suportada é ótima do respetivo *programa da soma pesada* das funções objetivo:

$$\max \sum_{k=1}^{p} \lambda_k f_k(x) \qquad (2)$$
$$\text{s. a: } x \in X$$

Em PLMO, quando o programa (2) é resolvido usando o método *simplex*, obtém-se sempre uma solução básica (vértice) eficiente do problema multiobjetivo. Soluções não básicas eficientes poderão ser conhecidas através do cálculo de ótimos alternativos para vetores de pesos particulares que conduzem às faces eficientes do problema. No caso dos problemas de PLIMO/PLIMMO, a otimização de uma soma pesada (2), usando o método *branch-and-bound*, conduz a uma solução eficiente suportada extrema. Se existirem ótimos alternativos, a exploração da árvore de *branch-and-bound* permite conhecer soluções eficientes suportadas não extremas. No entanto, nunca é possível obter soluções eficientes não suportadas com esta escalarização, mesmo que seja feita uma parametrização completa em λ e sejam exploradas todas as soluções alternativas para um dado vetor de pesos.

Abordaremos em seguida duas formas de escalarização que permitem caracterizar por completo o conjunto das soluções eficientes em problemas PLIMO ou PLIMMO: somas pesadas com restrições adicionais nas funções objetivo e escalarização baseada em pontos de referência.

2.1. Somas pesadas com restrições adicionais nas funções objetivo

Esta técnica consiste em introduzir restrições adicionais no programa da *soma pesada*, impondo limites inferiores para os valores das funções objetivo:

$$\max \sum_{k=1}^{p} \lambda_k f_k(x) \qquad (3)$$
$$\text{s. a: } f_k(x) \geq e_k,\ k=1,\ldots,p$$
$$x \in X$$

Esta técnica pode ser vista como uma particularização da escalarização genérica proposta por Soland (1979), em que a função escalar substituta $u[f(x)]$ pode ser arbitrária desde que cumpra alguns pressupostos de monotonicidade. A introdução de restrições nos valores das funções objetivo no programa da soma pesada permite calcular qualquer solução eficiente do problema, suportada ou não suportada. Para cada solução $x' \in X_E$, existe sempre algum vetor *e* tal que *x'* otimiza o programa (3) com $\lambda \in \Lambda$.

O tradicional método das restrições (*e-constraint*), em que se otimiza uma das funções objetivo passando as outras a restrições, pode também ser enquadrado neste tipo de escalarização. Neste caso, o peso atribuído à função escolhida para otimizar, seja $f_i(x)$, é 1, atribuindo-se peso 0 às outras (ou um escalar ε positivo suficiente pequeno, de modo a garantir que a solução obtida é eficiente e não apenas fracamente eficiente): $\max\ f_i(x) + \varepsilon \sum_{k \neq i} f_k(x)$. As restrições adicionais impõem limites inferiores para as *p*-1 funções objetivo diferentes da função *i*, isto é: $f_k(x) \geq e_k$,

k=1,…,p, $k \neq i$. Este processo de escalarização é também completo no sentido em que permite calcular qualquer solução eficiente do problema PLIMO/PLIMMO.

2.2 *Escalarização baseada em pontos de referência*

Podem ser definidos diferentes tipos de escalarização baseados em *pontos de referência*. Uma das formas habituais consiste na minimização da *métrica pesada de Chebyshev* ao ponto ideal ou a outro ponto de referência, cujos componentes representam geralmente níveis de aspiração que o decisor gostaria de atingir em cada função objetivo.

Considerando um ponto de referência $\bar{q} \in \Re^p$ que satisfaça $\bar{q} > f(x)$ para todo o $x \in X$, e pesos $w_k \geq 0$, $\forall k$, $\sum_{k=1}^{p} w_k = 1$, Bowman (1976) demonstrou que a parametrização completa em w de $\min_{x \in X} \left\| \bar{q} - f(x) \right\|_{\infty}^{w} = \min_{x \in X} \left(\max_{k=1,\ldots,p} \left\{ w_k \left| \bar{q} - f_k(x) \right| \right\} \right)$ permite calcular todas as soluções eficientes. Este programa pode também conduzir a soluções que são apenas fracamente eficientes, mas este resultado é evitado considerando a *métrica pesada e aumentada de Chebyshev*: $\min_{x \in X} \left\{ \left\| \bar{q} - f(x) \right\|_{\infty}^{w} - \rho \sum_{k=1}^{p} f_k(x) \right\}$, com ρ um escalar positivo bastante pequeno. O programa pode ser escrito da seguinte forma equivalente:

$$\begin{aligned} \min \quad & \left\{ \alpha - \rho \sum_{k=1}^{p} f_k(x) \right\} \\ \text{s.a:} \quad & w_k \left(\bar{q}_k - f_k(x) \right) \leq \alpha, \qquad k = 1,\ldots,p \\ & x \in X \\ & \alpha \geq 0 \end{aligned} \tag{4}$$

No caso de problemas de PLIMO, existe sempre um ρ suficientemente pequeno tal que qualquer solução eficiente do problema pode ser alcançada através do programa escalarizante (4). Já no que diz respeito a problemas de PLIMMO, podem existir pequenas regiões de soluções eficientes, junto a soluções fracamente eficientes, que o programa (4) não consegue alcançar. Um exemplo desta situação é apresentado em Alves e Clímaco (2001). No entanto, podemos admitir esta técnica como completa do ponto de vista prático, uma vez que é sempre possível escolher um ρ de tal forma pequeno que, para o decisor, as soluções 'escondidas' não se diferenciam das soluções vizinhas fracamente eficientes. Uma outra abordagem possível consiste em resolver o programa escalarizante de forma lexicográfica (Steuer e Choo, 1983): numa primeira fase, apenas α é minimizado; quando desta fase resultam ótimos alternativos, então é minimizado o segundo termo da função objetivo de (4) no conjunto das soluções ótimas da primeira fase de forma a eliminar as soluções fracamente eficientes. No entanto, este processo é computacionalmente mais dispendioso.

Existem outras técnicas baseadas em pontos de referência que permitem caracterizar por completo o conjunto das soluções eficientes de problemas PLIMO/PLIMMO. Uma abordagem comum consiste em não considerar pesos w_k (ou fixá-los em valores constantes) e variar o ponto de referência. Prova-se que, qualquer que seja a solução eficiente $x' \in X_E$, existem sempre pontos de referência para os quais x' otimiza o programa escalarizante (4) sem pesos. É ainda possível usar pontos de referência q que não satisfaçam a condição $q > f(x)$ $\forall x \in X$, desde que a variável α em (4) seja definida sem restrição de sinal ($\alpha \in \Re$). Neste caso, a minimização da função escalarizante poderá não corresponder à minimização de uma distância, porque o ponto de referência pode ser atingível, mas garante-se igualmente que é obtida uma solução eficiente do problema multiobjetivo. A função escalarizante pode ser interpretada como uma *função escalarizante de realização*: a minimização da função não significa "chegar perto" no sentido tradicional, mas "chegar perto ou ultrapassar" (Wierzbicki, 1998).

3. Métodos para PLIMO/PLIMMO

Os métodos desenvolvidos para apoio à decisão em problemas de PLIMO/PLIMMO podem classificar-se em métodos *geradores* – desenhados para determinar todas as soluções eficientes do problema ou um subconjunto pré-definido destas soluções (por exemplo, todas as soluções suportadas ou só as extremas) e métodos *interativos* – que se caracterizam por fases de cálculo alternadas com fases de interação, nas quais o decisor expressa as suas preferências que são usadas na definição de parâmetros para a fase de cálculo seguinte.

Os métodos geradores exigem normalmente um esforço computacional elevado, não só por calcularem um número elevado de soluções eficientes mas, sobretudo, para assegurarem o requisito de todas as soluções serem calculadas. Por esta razão, observa-se que a maior parte dos métodos geradores se destinam a problemas apenas com variáveis binárias (pela maior facilidade em usar técnicas de enumeração) ou problemas bi-objetivo. Vários destes métodos foram desenvolvidos nas décadas de 70 e 80, observando-se nas décadas seguintes um maior desenvolvimento de métodos interativos. Há, porém, alguns desenvolvimentos em trabalhos muito recentes.

Ao longo das últimas décadas têm sido publicados vários artigos de revisão da literatura, em que são feitos levantamentos de métodos para problemas multiobjetivo com variáveis inteiras: Teghem e Kunsch (1986) fazem uma revisão de métodos interativos para PLIMO e PLIMMO publicados até ao final de 1985; cobrindo o mesmo período de tempo, Rasmussen (1986) apresenta uma revisão de métodos interativos e não-interativos para programação 0-1 multiobjetivo; Clímaco et al. (1997) apresentam uma classificação de métodos de programação inteira multiobjetivo, linear e não linear. Mais recentemente, Alves e Clímaco (2007) publicaram uma revisão de métodos interativos de PLIMO/PLIMMO, onde são caraterizados e resumidos cerca de vinte métodos interativos.

Neste texto focar-nos-emos nos métodos geradores, apresentando uma visão geral das metodologias existentes e das limitações e dificuldades que enfrentam. Sem intenção de exaustividade, procuraremos mostrar métodos ilustrativos de diferentes estratégias, começando com algoritmos destinados a problemas de programação 0-1 multiobjetivo.

Bitran (1977, 1979), Kiziltan e Yucaoglu (1983) e Deckro e Winkofsky (1983) propõem algoritmos de enumeração implícita para problemas de PLIMO com variáveis binárias. Bitran (1977) começa por analisar o problema multiobjetivo 0-1 sem restrições funcionais. Com base no princípio de que soluções eficientes do problema sem restrições que sejam admissíveis para o original são soluções eficientes do problema original, propõe um algoritmo que se desenvolve em três passos: *i*) caracterizar por completo o conjunto eficiente do problema sem restrições; *ii*) determinar, de entre o conjunto anterior, as soluções que são admissíveis para o problema original; *iii*) acrescentar soluções, examinando outros pontos (não obtidos em *i*) a partir de direções de preferência ao longo das quais os objetivos podem ser melhorados. É este o passo mais delicado do algoritmo, envolvendo um grande esforço computacional. Em Bitran (1979) é apresentada um processo mais eficaz do ponto de vista computacional.

O algoritmo de Kiziltan e Yucaoglu (1983) é uma extensão para o caso multiobjectivo do conhecido algoritmo de Balas (1965) para programação 0-1 e gera todas as soluções eficientes do problema multiobjectivo. Também Deckro e Winkofsky (1983) geram todas as soluções eficientes usando um algoritmo de enumeração implícita. O algoritmo vai gerando soluções candidatas a soluções eficientes e, só no final, é que se sabe as que são realmente eficientes. Esta é também uma característica do método de Kiziltan e Yucaoglu (1983). Estes métodos, que operam com soluções *potencialmente* eficientes durante as fases intermédias do processo, não podem ser interrompidos a meio do processo, sob pena de devolverem soluções que não são eficientes. Já o algoritmo de Bitran (1977, 1979) usa um processo *construtivo*, em que novas soluções eficientes vão sendo sucessivamente geradas e adicionadas ao conjunto das soluções eficientes. Este tipo de métodos geradores pode ser interrompido antes de chegar ao fim, devolvendo um subconjunto das soluções eficientes.

Passando a métodos que se aplicam a problemas PLIMO com variáveis inteiras genéricas, e seguindo uma ordem cronológica, começamos por referir o trabalho de Villareal e Karwan (1981). Os autores propõem duas abordagens para gerar todas as soluções eficientes de um problema multiobjectivo inteiro puro com variáveis limitadas. A primeira abordagem usa programação dinâmica e a segunda combina programação dinâmica com *branch-and-bound*. Estes algoritmos também trabalham com soluções *potencialmente* eficientes, em que só no final se conhecem as soluções verdadeiramente eficientes.

Klein e Hannan (1982) usam um processo construtivo em que se vai restringindo a região admissível através de restrições auxiliares que eliminam as soluções eficientes já calculadas e soluções dominadas por estas. Estas restrições impõem que a solução seguinte tem que ser melhor em *alguma* função objetivo (condições '$\vee$') relativamente a *todos* os pontos não dominados já conhecidos (condições '$\wedge$'). Como a formalização das condições '$\vee$' é feita à custa de variáveis binárias auxiliares, a dimensão do programa auxiliar aumenta de iteração para iteração. O processo termina quando a região admissível do programa escalar se torna vazia. Sylva e Crema (2004) apresentam uma variação deste algoritmo em que é maximizada uma soma pesada das funções objetivo em cada iteração, em vez de apenas um dos objetivos.

Importa referir que todos os métodos anteriores se aplicam apenas a problemas inteiros puros. Mavrotas e Diakoulaki (1998) propuseram um método gerador para o caso 0-1 misto. A técnica consiste em gerar, através de um algoritmo de *branch-and*-bound, todas as soluções *potencialmente* não dominadas e ir eliminando sucessivamente as soluções dominadas através de comparações par a par, até que no final restem apenas as eficientes. Os resultados computacionais apresentados neste artigo ilustram bem o esforço computacional envolvido na geração completa do conjunto eficiente, de tal forma que autores sugerem a intervenção do decisor para restringir o âmbito da pesquisa, propondo um procedimento interativo.

As dificuldades do cálculo exaustivo das soluções eficientes, tanto em problemas inteiros puros como em problemas inteiros mistos, reduzem-se consideravelmente no caso bi-objetivo. Uma abordagem semelhante à de Klein e Hannan (1982) é proposta por Chalmet et al. (1986) para problemas de PLI bi-objetivo, assumindo que as funções objetivo só tomam valores inteiros em X. Utilizam a escalarização (3) em que restrições nas funções objetivo são baseadas nos pontos não dominados já conhecidos. O processo consiste em estabelecer pares de soluções candidatas a adjacentes, (z^a, z^b), e analisar se existe alguma solução não dominada entre elas resolvendo: max $\{\lambda_1 f_1(x) + \lambda_2 f_2(x)$: $x \in X$, $f_k(x) \geq \breve{z}_k + 1$, $k=1,2\}$ com $\breve{z}_k = \min\{ z_k^a, z_k^b \}$. Solanki (1991) adota igualmente o princípio de exploração de soluções intermédias entre soluções eficientes candidatas a adjacentes, mas usa a métrica pesada de Chebyshev (com alteração do ponto de referência e dos pesos em cada iteração). O método aplica-se a problemas de programação linear inteira-mista bi-objetivo (PLIMBO) e procura gerar um subconjunto representativo (isto é, bem distribuído) de soluções não dominadas, considerando, para tal, um *erro* máximo que define a condição de paragem do algoritmo. Quanto menor for o erro, mais o algoritmo se aproxima de um algoritmo gerador completo.

Também o método de Alves e Clímaco (2000), desenvolvido como método interativo para PLIMMO, pode ser usado como método *gerador* em PLIMBO. Este método usa o programa escalarizante da métrica aumentada de Chebychev, parametrizado no ponto de referência. O método é especialmente vocacionado para *pesquisas direcionais*, alterando, em cada iteração, apenas a componente do ponto de referência correspondente à função objetivo que se pretende melhorar naquele momento. O método incorpora técnicas de análise de sensibilidade que alteram de forma automática o ponto de referência. Calcula, assim, a solução não dominada '*mais próxima*' da anterior segundo a pesquisa direcional que está a prosseguir. No caso bi-objetivo, se a pesquisa se iniciar no ótimo de uma das funções, e a outra função for escolhida para melhorar, o método varre todas as soluções não dominadas, suportadas e não suportadas. No caso inteiro-misto, usa um *passo* (que pode ser tão pequeno quanto ser quiser) para determinar o espaçamento máximo de soluções contínuas calculadas, saltando automaticamente as descontinuidades.

Por fim, é também simples a configuração do tradicional método das restrições (*e-constraint*) para calcular todas as soluções não dominadas de um problema inteiro puro bi-objectivo: considere-se a maximização de $f_1(x) + \varepsilon f_2(x)$, (com ε muito pequeno, apenas para garantir eficiência da solução obtida), restringindo $f_2(x) \geq e_2$ em $x \in X$; começando por calcular a solução extrema que otimiza f_1, e assumindo que as funções só tomam valores inteiros, a solução seguinte obtém-se considerando $e_2 = f_2(x^*)+1$, com x^* a solução ótima do programa escalarizante anterior; o processo termina quando a região admissível for vazia.

Özlen e Azizoğlu (2009) propõem uma extensão para 3 funções objetivo do método das restrições descrito acima para problemas inteiros puros bi-objetivo. Consideram o seguinte problema bi-objetivo: $\{\max\ f_1(x) + \varepsilon f_3(x), \max\ f_2(x) + \varepsilon f_3(x) : x \in X,\ f_3(x) \geq e_3\}$. Para gerar o conjunto não dominado do problema tri-objetivo, começam por atribuir a e_3 um valor suficientemente pequeno que não corte a região admissível e geram todas as soluções não dominadas do problema bi-objetivo; depois aumentam e_3 de forma sistemática ($e_3 = \min\{f_3(x): x \in BE^*\}+1$, com BE^* o conjunto eficiente do problema bi-objetivo anterior) e repetem o processo, de forma a gerar todas as soluções não dominadas.

Przybylski et al. (2010a) propõem um algoritmo para determinar todos os pontos não dominados suportados extremos em problemas tri-objetivo. Przybylski et al. (2010b) estendem estre trabalho, apresentando um método de duas fases, em que a primeira fase calcula as soluções suportadas (usando o algoritmo anterior) e a segunda fase calcula as soluções não suportadas, usando métodos enumerativos. Tanto nestes trabalhos, como no de Özlen e Azizoğlu (2009), são ainda propostas generalizações para mais do que três funções objetivo. No entanto, o seu interesse é essencialmente teórico, uma vez que poderão ser difíceis de implementar e computacionalmente muito exigentes dada a recursividade dos algoritmos.

Na sequência dos desenvolvimentos de Przybylski et al. (2010a), Özpeynirci e Köksalan (2010) propõem um outro algoritmo para determinar todas soluções não dominadas suportadas extremas em PLIMMO. A ideia chave destes dois trabalhos baseia-se na exploração do espaço dos pesos para a otimização de somas pesadas das funções objetivo. Também nós temos desenvolvido trabalho neste âmbito para problemas tri-objetivo. Estes trabalhos são apresentados com maior detalhe na próxima secção.

4. Novos desenvolvimentos em PLIMMO baseados na exploração do espaço dos pesos

Considere-se o processo de escalarização da *soma pesada* das funções objetivo (2) e o conjunto dos vetores de pesos, vulgarmente designado por *espaço dos pesos*, definido por $\Lambda=\{ \lambda \in \Re^p:\ \lambda_k > 0,\ k=1,2,...,p,\ \sum_{k=1}^{p} \lambda_k = 1\}$. O espaço dos pesos pode ser decomposto em subconjuntos $\Lambda(z')$ para cada $z' \in Z_{ND}$ *suportada* (note-se que apenas as soluções *suportadas* têm representação no espaço dos pesos). A cada um destes subconjuntos chama-se *região de indiferença* no espaço dos pesos porque todos os vetores de $\Lambda(z')$ conduzem à mesma solução z'. Assim, $\Lambda(z')=\{\lambda \in \Lambda: \lambda z' \geq \lambda z$ para todo o $z \in Z_{ND}\}$. As regiões de indiferença são sempre convexas (polítopos convexos). Além disso, os pontos *não dominados suportados extremos* (*NDSE*) permitem a decomposição completa do espaço dos pesos, porque estes pontos, e apenas estes, correspondem a regiões de indiferença de dimensão p-1 (a mesma dimensão de Λ); aos pontos não extremos estão associadas regiões de indiferença que resultam da interseção de regiões de indiferença de pontos extremos (demonstrações destas propriedades podem ser vistas em Przybylski et al., 2010a).

Designemos por Z_{NDSE} o conjunto de todos os pontos *NDSE*. O algoritmo proposto por Przybylski et al. (2010a) para determinar Z_{NDSE} baseia-se no seguinte princípio: em cada iteração, o espaço dos pesos é decomposto por completo com as soluções conhecidas até ao momento, seja esse conjunto S, criando *super-regiões*: $\Lambda^+(z')=\{\lambda \in \Lambda: \lambda z' \geq \lambda z$ para todo o $z \in S\}$. Para cada ponto $z' \in S$, o algoritmo procura novos pontos nas fronteiras de $\Lambda^+(z')$ e atualiza a decomposição do espaço dos pesos conforme vão sendo adicionados novos pontos a S. Por exemplo, no caso tri-objetivo, se z', $z'' \in S$ são adjacentes, então $\Lambda^+(z') \cap \Lambda^+(z'')$ é um segmento de reta – sejam λ^1, λ^2 os

pontos extremos deste segmento de reta. O algoritmo procura novas soluções no segmento λ^1,λ^2 calculando todos os pontos *NDSE* do seguinte problema bi-objetivo: {max $f_1'(x)=\lambda^1 Cx$, max $f_2'(x)=\lambda^2 Cx : x \in X$}. Define-se, desta forma, um algoritmo recursivo.

Özpeynirci e Köksalan (2010) propõem um outro algoritmo, com o mesmo propósito, que não usa recursividade mas tem uma forte componente combinatória. A ideia base consiste na introdução de p pontos *fictícios* no espaço dos objetivos, $Z_m=\{m^k, k=1,\ldots,p\}$, não admissíveis e não dominados em relação a todos os pontos de Z_{NDSE}. Estes pontos têm características tais que as suas regiões de indiferença ocupam toda a fronteira do espaço dos pesos (isto é, quando algum peso se aproxima de zero). Os pontos m^k são incorporados no conjunto de pesquisa, que passa a ser $Z_{NDSEm} = Z_{NDSE} \cup Z_m$. Os autores provam que cada ponto $z' \in Z_{NDSEm}$ é adjacente a pelo menos p pontos em Z_{NDSEm}. Tal como anteriormente, seja S o conjunto de pontos *NDSE* conhecidos até ao momento. O algoritmo seleciona sub-conjuntos de p pontos de $S \cup Z_m$ e, para cada um, define o vetor λ normal ao hiperplano que passa por esses pontos. Se todas as componentes de λ forem positivas, então é otimizada a respetiva soma pesada para tentar encontrar um novo ponto *NDSE*.

Nos algoritmos anteriores, não ficamos a conhecer nas fases intermédias as regiões de indiferença das soluções calculadas. Esta informação fica disponível apenas no final, quando se conhecem todas as soluções *NDSE*. As regiões de indiferença fornecem informação importante ao decisor, uma vez que ele pode ser indiferente a todas as combinações de pesos nessa região dado que todas elas conduzem à mesma solução não dominada. No sentido de obter essa informação, desenvolvemos um processo que permite calcular uma *sub-região de indiferença* de cada vez que é calculada uma solução *NDSE*. A partir da união de sub-regiões, e da exploração gráfica para p=3, desenvolvemos algoritmos que permitem gerar todas as soluções *NDSE* adjacentes a uma solução ou gerar todas as soluções *NDSE* de um problema de PLIMO/PLIMMO com três funções objetivo.

Em PLMO, a região de indiferença de uma solução básica eficiente é definida por $\{\lambda \in \Lambda: \lambda W \geq 0\}$, com W a matriz dos custos reduzidos dada por $W=C_B B^{-1}N-C_N$ em que: B e N, C_B e C_N são sub-matrizes de A e de C correspondentes às variáveis básicas (x_B) e não básicas (x_N), respetivamente. Considere-se agora um problema com variáveis inteiras (PLIMO/PLIMMO), em que foi otimizada uma soma pesada das funções objetivo usando o método *branch-and-bound*, obtendo-se a solução eficiente cujo ponto não dominado é z^o. Uma sub-região de indiferença desta solução, seja $\overline{\Lambda}(z^o)$, pode ser calculada da seguinte forma:

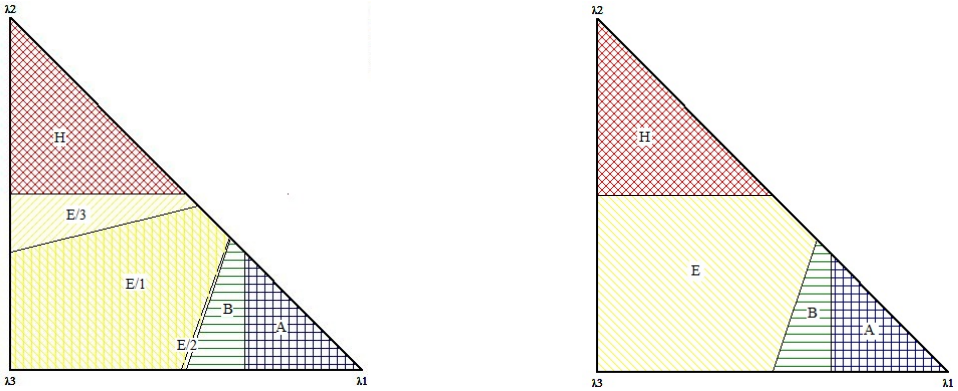
a) Calculam-se as regiões de indiferença $R^{(i)}$ (tal como em PLMO) associadas a cada nó terminal i da árvore de *branch-and-bound* cujo problema PL é admissível (conjunto de índices T) e considera-se a sua intersecção: $R=\bigcap_{i\in T} R^{(i)}$

b) Acrescentam-se as restrições resultantes da comparação do valor agregado das funções objetivo no nó ótimo (nó o) em relação aos outros nós terminais da árvore cujo problema PL é admissível: $\overline{\Lambda}(z') = R \cap \{\lambda z^o \geq \lambda z^{(i)}$, para todo $i \in T\backslash\{o\}\}$.

Tratando-se de problemas tri-objetivo, estas regiões podem ser representadas graficamente, como se ilustra no exemplo seguinte.

Exemplo 2: Considere-se o problema do *exemplo* 1 com mais uma função objetivo: $f_3(x)=x_1+2x_2$. As soluções *NDSE* deste problema tri-objetivo são A, B, E e H (fig. 1), sendo A, H, E as soluções que otimizam individualmente f_1, f_2 e f_3 respetivamente. A figura 3(a) mostra sub-regiões de indiferença calculadas segundo o processo anterior a partir da otimização de diferentes somas pesadas: a otimização de somas pesadas com combinações de pesos extremas conduziu às regiões A, H (completas) e E/1 (sub-região de E). Escolhendo, em seguida, um vetor de pesos entre A e E/1, fomos conduzidos a B em que a região calculada é completa. Escolhendo depois um vetor de pesos na pequena faixa entre B e E/1, a solução obtida foi novamente E com a sub-região de indiferença E/2 que inclui, não só essa faixa, como toda a área E/1. Por fim, da escolha de um vetor de pesos entre E/1 e H, obteve-se a sub-região E/3 (que apenas interseta as outras sub-regiões de E na fronteira). A figura 3(b) mostra a descomposição completa do triângulo depois da

agregação das sub-regiões obtidas anteriormente. Estas figuras foram retiradas do sistema computacional que desenvolvemos para problemas com 2 ou 3 funções objetivo.



(a) Sub-regiões de indiferença (b) Regiões de indiferença

Fig. 3 – Decomposição do espaço dos pesos do problema do *Exemplo* 2.

É de salientar que as regiões de indiferença são *convexas*, o que permite a agregação de sub-regiões já conhecidas da mesma solução num invólucro convexo que é, garantidamente, uma sub-região de indiferença dessa solução.

O algoritmo para calcular todas as soluções *NDSE* adjacentes a uma dada solução, em problemas tri-objetivo, baseia-se na exploração das arestas que definem o polígono da sub-região de indiferença dessa solução. Para cada aresta ainda *não explorada*, que não esteja na fronteira do triângulo, é selecionado um vetor de pesos exterior ao polígono, muito próximo e central relativamente à aresta em análise (ou seja, as componentes λ_1,λ_2 do vetor selecionado distam ε, um valor bastante pequeno, das respetivas componentes do ponto médio da aresta em análise). É então otimizada a respetiva soma pesada das funções objetivo, da qual podem resultar duas situações:

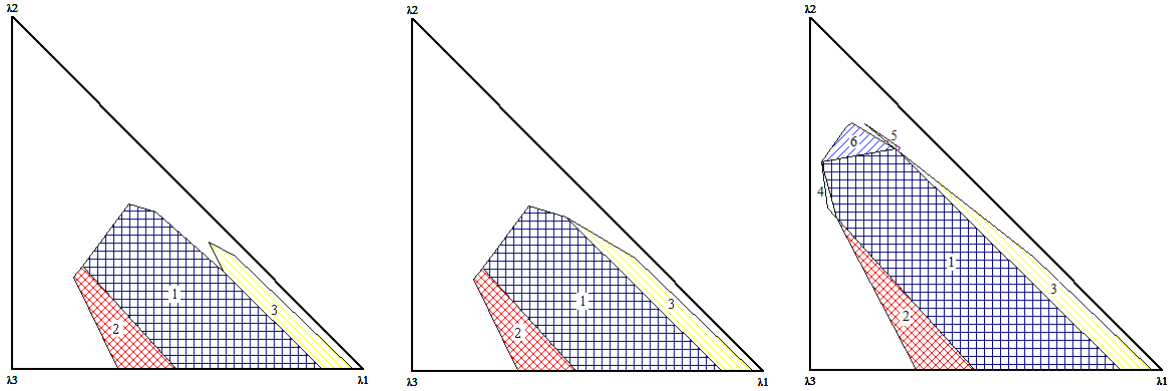
i) é calculada novamente a mesma solução – neste caso, agregam-se as sub-regiões de indiferença, marcando como *não exploradas* as novas arestas criadas;

ii) é encontrada uma nova solução eficiente, marcando-se a aresta como *explorada*.

Neste último caso, podem acontecer situações como a ilustrada na fig. 4(a) na passagem da solução 1 para a 3, em que a sub-região de indiferença da nova solução (sol. 3) não toca a aresta em análise (da sol. 1) em toda a sua extensão. A seguinte propriedade (cuja prova se omite aqui) é fundamental para o bom desempenho do algoritmo.

Propriedade: sejam λ^0, λ^1 e λ^2 três vetores de pesos sobre a mesma reta, em que λ^0 e λ^1 pertencem às regiões de indiferença de duas soluções; se λ^2 pertence à região de indiferença de uma delas, então λ^2 também pertence à região de indiferença da outra solução.

Esta propriedade permite a expansão da região da nova solução, tal como é ilustrada na fig. 4(b). O inverso também pode acontecer, ou seja a nova região pode ter uma aresta comum mais longa do que a aresta em análise. Nesse caso, será a primeira solução que vê a sua região expandida. Na implementação computacional do algoritmo foram ainda incluídos outros melhoramentos relacionados com a expansão de sub-regiões de indiferença. Por exemplo, a situação (*i*), em que há um aumento da sub-região em análise por agregação, pode conduzir (pela *propriedade* anterior) a expansões de sub-regiões adjacentes. A figura 4(c) mostra a expansão das regiões 2 e 3 por consequência do alargamento de 1, bem como todas as sub-regiões de indiferença obtidas pelo cálculo das soluções adjacentes à solução 1.



(a) (b) (c)

Fig. 4 – Cálculo de soluções adjacentes e expansão de sub-regiões de indiferença

A extensão deste algoritmo para um algoritmo *gerador* de todas as soluções *NDSE* foi também implementada: começando com uma qualquer solução *NDSE*, são calculadas todas as soluções que lhe são adjacentes, repetindo-se o processo para cada uma das adjacentes. Neste caso, sempre que é encontrada uma nova solução (situação *ii*), é marcada como *explorada* não só a aresta em análise, como também a aresta coincidente da nova região de indiferença.

5. Considerações finais

Neste artigo apresentámos os conceitos fundamentais em PLIMO e PLIMMO. Também revimos os principais métodos para PLIMO/PLIMMO, com particular destaque para métodos *geradores*. Por fim, referimos desenvolvimentos recentes na área baseados na exploração do espaço dos pesos, incluindo algum trabalho realizado por nós.

Agradecimento

Este trabalho foi parcialmente apoiado pela FCT no âmbito do projeto PEst-C/EEI/UI0308/2011.

Referências

Alves, M.J. e Clímaco, J., (2000), An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound, *European Journal of Operational Research*, 124 (3), 478-494.

Alves, M.J. e Clímaco, J., Multiobjective mixed-integer programming, em Pardalos, P. M. e Floudas, C. A. (Eds.), *Encyclopedia of Optimization*, vol. III. Kluwer Academic Publishers, 466-472, 2001.

Alves, M.J. e Clímaco, J. (2007), A review of interactive methods for multiobjective integer and mixed-integer programming, *European Journal of Operational Research*, 180, 99-115.

Balas, E. (1965), An additive algorithm for solving linear problems with zero-one variables. *Operations Research*, 15, 517-546.

Bitran, G. R. (1977), Linear multiple objective programs with zero-one variables, *Mathematical Programming*, 13, 121-139.

Bitran, G. R. (1979), Theory and algorithms for linear multiple objective programs with zero-one variables, *Mathematical Programming*, 17 (3), 362-389.

Bowman Jr, V. J., On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives, em Thiriez, H. e Zionts. S. (Eds), *Multiple Criteria Decision Making*, Lecture Notes in Economics and Mathematical Systems, 130, Springer-Verlag, Berlin, 76-86, 1976.

Chalmet, L. G., Lemonidis, L. e Elzinga, D. J. (1986), An algorithm for the bi-criterion integer programming problem, *European Journal of Operational Research*, 25, 292-300.

Clímaco, J., Ferreira, C. e Captivo, M. E., Multicriteria integer programming: an overview of the different algorithmic approaches, em Clímaco, J. (Ed.), *Multicriteria Analysis*, Springer-Verlag, Berlin, 248-258, 1997.

Deckro, R. F. e Winkofsky, E. P. (1983), Solving zero-one multiple objective programs through implicit enumeration, *European Journal of Operational Research*, 12, 362-374.

Kiziltan, G. e Yucaoglu, E. (1983), An algorithm for multiobjective zero-one linear programming, *Management Science*, 29 (12), 1444-1453.

Klein, D. e Hannan, E. (1982), An algorithm for the multiple objective integer linear programming problem, *European Journal of Operational Research*, 9, 378-385.

Mavrotas, G. e Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming, *European Journal of Operational Research*, 107, 530-541.

Özlen, M. e Azizoğlu, M. (2009), Multi-objective integer programming: A general approach for generating all non-dominated solutions, *European Journal of Operational Research*, 199, 25-35.

Özpeynirci, O. e Köksalan, M. (2010), An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer problems, *Management Science*, 56 (12), 2302-2315.

Przybylski, A., Gandibleux, X. e Ehrgott, M. (2010a), A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme, *INFORMS Journal on Computing*, 22(3), 371-386.

Przybylski, A., Gandibleux, X. e Ehrgott, M. (2010b), A two-phase method for multi-objective integer programming and its application to the assignment problem with three objetives, *Discrete Optimization*, 7, 149-165.

Rasmussen, L. M. (1986), Zero-one programming with multiple criteria, *European Journal of Operational Research*, 26, 83-95.

Soland, R. M. (1979), Multicriteria optimization: a general characterization of efficient solutions, *Decision Sciences*, 10, 26-38.

Solanki, R. (1991). Generating the noninferior set in mixed integer biobjective linear programs: an application to a location problem, *Computers and Operations Research*, 18 (1), 1-15.

Steuer, R.E. e Choo, E.-U (1983), An interactive weighted Tchebycheff procedure for multiple objective programming, *Mathematical Programming*, 26, 326-344.

Sylva, J. e Crema, A. (2004), a method for finding the set of nondominated vectors for multiple objective integer linear programs, *European Journal of Operational Research*, 158, 46-55.

Teghem, J. e Kunsch, P. L., Interactive methods for multi-objective integer linear programming, em Fandel, G. et al. (Eds), *Large-Scale Modelling and Interactive Decision Analysis*, Lecture Notes in Economics and Mathematical Systems, 273, Springer-Verlag, Berlin, 75-87, 1986.

Villareal, B. e Karwan, M. H. (1981), Multicriteria integer programming: A (hybrid) dynamic programming recursive approach, *Mathematical Program*ming, 21, 204-223.

Wierzbicki, A. P., Reference point methods in vector optimization and decision support, *Interim Report IR-98-017*, IIASA, Laxenburg, Austria, 1998.