Localização de Postos de Vigia de Fogos Florestais e Outras Coisas

Jorge Orestes Cerdeira
Universidade Nova de Lisboa
jo.cerdeira@fct.unl.pt

Onde instalar postos para vigia de fogos florestais? Como desenhar uma rede de áreas para proteção da biodiversidade? Como selecionar candidatos a guias, em idiomas diferentes, para um museu? Vamos ver que estas questões não são mais do que diferentes contextualizações do mesmo problema e que poderá não ser exequível encontrar as soluções mais eficientes.

No caso da instalação de postos de vigia, são dados o conjunto das parcelas florestais que se pretende monitorizar relativamente à ocorrência de fogos, os locais onde é possível a instalação de postos de vigia e a indicação, para cada um desses locais, das parcelas que ficam sob vigilância caso seja instalado um posto nesse local. Quer-se decidir onde instalar postos de forma a monitorizar todas as parcelas florestais e, como há *custos* decorrentes da construção e da utilização de cada posto de vigilância, pretende-se que o número de postos a instalar seja mínimo.

Para o traçado de redes de áreas prioritárias para a conservação, são conhecidos o conjunto de parcelas em que se divide a área de estudo, as espécies dessa região com interesse para a conservação e as espécies representadas em cada uma das parcelas. O problema básico na identificação de áreas prioritárias para a conservação consiste em descobrir um conjunto de parcelas, em número mínimo, em que estejam representadas todas as espécies.

Finalmente, para selecionar candidatos a guias, em idiomas diferentes, para um museu, após estabelecidos os idiomas desejados e conhecidos os idiomas que cada candidato domina, procede-se à escolha de um conjunto de candidatos que, com o menor número possível, assegurem visitas guiadas em todos os idiomas.

Os problemas anteriores podem ser descritos usando diagramas como o da figura 1, em que estão representados pontos agrupados em dois conjuntos (não vazios) $E = \{1, 2, \cdots, 7\}$ e $D = \{8, 9, \cdots, 16\}$, e algumas ligações entre pares de pontos em que um dos pontos está em E e o outro em D. O objeto assim representado chama-se *grafo bipartido* e E, D são as classes de bipartição.

Os pontos de E representam (i) os locais onde poderão ser instalados postos de vigia de fogos, (ii) as parcelas da área de estudo, ou (iii) os candidatos a guias. Os pontos de D representam (i') as parcelas florestais a monitorizar, (ii') as espécies a proteger, ou (iii') os idiomas que se pretende usar nas visitas guiadas. A existência da ligação $[u, v]$, com $u \in E$ e $v \in D$, indica que (i'') a instalação de um posto no local u permite vigiar a ocorrência de fogos na parcela florestal v, (ii'') a espécie v está representada na parcela u, ou (iii'') o candidato u fala o idioma v.

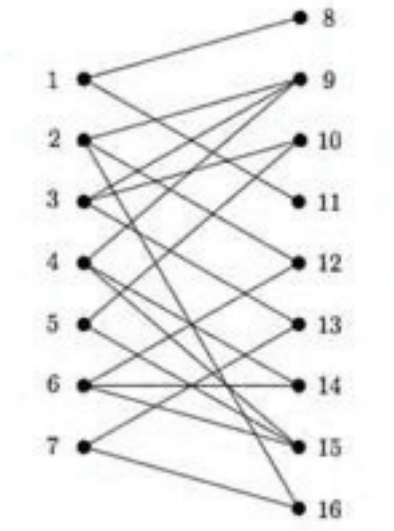

Figura 1: Grafo bipartido com classes de bipartição $E = \{1,2,\cdots 7\}$ e $D = \{8,9,\cdots ,16\}$

Um grafo bipartido pode ser adequadamente codificado para ser introduzido num computador como uma matriz de 0's e 1's, do tipo $|E| \times |D|$, em que o elemento (u, v) da matriz é 1 (0) se (não) existe ligação entre $u \in E$ e $v \in D$.

Uma *solução admissível* para cada um dos problemas anteriores é um conjunto C de pontos de E tal que todo o ponto de D esteja ligado a pelo menos um ponto em C. Diz-se que C é uma *cobertura* (de D). Os conjuntos {1, 4, 5, 6, 7} e {1, 2, 3, 6} são coberturas (dos pontos do lado direito) do grafo da figura 1, com cardinalidades 5 e 4, respetivamente. Os problemas da

instalação de postos de vigia contra fogos florestais, da identificação de áreas prioritárias para a conservação e da seleção de candidatos para guias em idiomas diferentes num museu procuram *coberturas mínimas*, i.e., de menor cardinalidade. O leitor não terá dificuldade em verificar que o conjunto {1, 2, 3, 6} é uma cobertura mínima (dos pontos do lado direito) do grafo da figura 1.

Coloca-se pois, a questão de determinar coberturas mínimas.

Poderíamos pensar procurar coberturas mínimas por enumeração. Por exemplo, começando com um inteiro arbitrário k ($1 \leq k \leq |E|$), repetidamente enumerando os subconjuntos de E com cardinalidade k, e atualizando o valor de k, diminuindo-o ou aumentando-o caso um dos subconjuntos seja ou não cobertura. O procedimento termina quando for encontrada uma cobertura C e nenhum subconjunto de cardinalidade $|C|-1$ for cobertura. Mas será que este procedimento é exequível? Suponhamos que é possível decidir se um subconjunto arbitrário de E é cobertura num nanossegundo (10^{-9} segundos), que é o valor aproximado do tempo de execução de uma *operação elementar* (uma comparação, uma atribuição de valor, uma adição, ...) num computador atual. Para $|E| = 10$ e $k = 3$, testar se cada um dos 120 subconjuntos de cardinalidade 3 de E é cobertura demoraria cerca de 0.00000012 segundos. Mas 17 séculos seriam insuficientes se $|E| = 70$ e $k = 30$. Tendo em conta que os problemas de determinação de coberturas mínimas que se colocam em situações reais incidem, normalmente, sobre dados de grandes dimensões (digamos, $|E|$ e $|D|$ nas ordens das várias centenas e vários milhares, respetivamente), métodos de enumeração como o anterior são completamente impraticáveis.

Há pois que considerar estratégias mais eficientes. Uma possibilidade, aparentemente razoável, seria, em cada iteração, adicionar a um subconjunto $C \subset E$ um elemento u de $E \setminus C$ com o maior número de vizinhos ainda não cobertos por C. Esta estratégia, que em cada momento opta por um agente que garante o maior retorno imediato, chama-se *gulosa*. Vejamos como opera o algoritmo guloso com o grafo bipartido da figura 2, em que $E = \{a, b, c, d, e\}$ e $D = \{f, g, h, i, j, k\}$.

O primeiro ponto a ser incluído em C é o ponto a, uma vez que tem o maior número de vizinhos (4). Ficam por cobrir j e k. Cada ponto de $E \setminus \{a\}$ cobre exatamente um desses dois pontos. Assim, C é ampliado com um qualquer ponto diferente de a. Vamos supor que é escolhido d, restando ainda por cobrir o ponto k. Os candidatos a inclusão em C são c e e, e uma vez incluído um qualquer desses pontos, o algoritmo termina com a obtenção de uma cobertura de cardinalidade 3. No entanto a cobertura obtida não é mínima, uma vez que $\{b, c\}$ é também cobertura.

Desta vez o algoritmo guloso falhou na determinação da solução ótima. Será sempre assim? Será que não há garantia da otimalidade das soluções gulosas dos problemas de otimização combinatória? Para responder a esta questão comecemos por definir problema de otimização combinatória.

Seja A um conjunto finito e $c_e \geq 0$ um *custo* associado a todo o elemento e de A. Se X é um subconjunto de A, dizemos que o custo de X é a soma dos custos dos elementos em X, que denotamos por $c(X)$. Consideremos um conjunto S de subconjuntos de A. Aos elementos de S chamamos *soluções admissíveis*. O *problema de otimização combinatória* (POC) associado ao terno (A, c, S) consiste na determinação de uma solução admissível de custo máximo ou mínimo.

No problema da cobertura mínima, A é o conjunto dos pontos de uma das classes de bipartição do grafo, $c_e = 1$, qualquer que seja o ponto e de A e o conjunto das soluções admissíveis S é o conjunto das coberturas.

Em muitos POC o conjunto S é fechado para a inclusão, i.e., ao eliminar elementos arbitrários de uma qualquer solução admissível obtém-se uma solução admissível. Curiosamente, é precisamente o oposto do que acontece com o problema da cobertura. Obtém-se uma cobertura ao incluir um qualquer ponto numa cobertura. No entanto, podemos facilmente transformar o problema da cobertura num POC equivalente cujo conjunto de soluções admissíveis é fechado

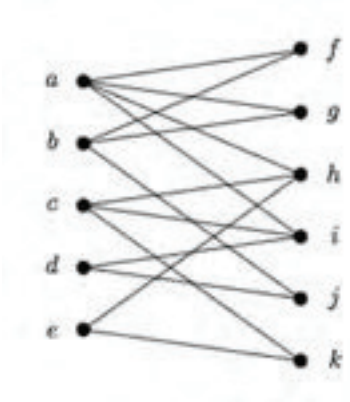

Figura 2: Grafo bipartido.

para a inclusão. Basta considerar que as soluções admissíveis são os complementares das coberturas, i.e., $X \subset A$ é admissível sse $A \setminus X$ é cobertura. Uma solução admissível de custo máximo X^* deste problema identifica uma cobertura $A \setminus X^*$ de custo mínimo.

Quando S é fechado para a inclusão diz-se que o par (A, S) é um *sistema de independência* e os conjuntos de S (de $2^A \setminus S$) chamam-se *independentes* (*dependentes*).

Para obter uma solução admissível de custo máximo do POC associado a um sistema de independência (A, S), com custos não negativos definidos sobre os elementos de A, o algoritmo guloso, em cada iteração, insere em X, que inicialmente é o conjunto vazio, um elemento de maior custo $e \in A \setminus X$, tal que $X \cup \{e\} \in \mathcal{S}$. Se não existe e $e \in A \setminus X$ com $X \cup \{e\} \in \mathcal{S}$, o algoritmo para e devolve o independente *maximal* X.

Q 1. *Em que condições o algoritmo guloso resolve corretamente o POC associado a um sistema de independência (A,S), i.e., determina um independente de custo máximo, quaisquer que sejam os custos não negativos atribuídos aos elementos do conjunto A?*

A resposta a esta questão é dada pelo resultado seguinte:

R 1. *O algoritmo guloso resolve corretamente o POC associado ao sistema de independência M = (A, S) se e só se M é um matróide.*

Um sistema de independência $M = (A,S)$ é um *matróide* se $X, Y \in \mathcal{S}$, com $|Y| > |X|$, então $X \cup \{y\} \in \mathcal{S}$, para algum $y \in Y \setminus X$. Por outras palavras, dados dois independentes com cardinalidades diferentes, é possível ampliar o de menor cardinalidade com algum elemento do de maior cardinalidade, mantendo independência.

Vejamos exemplos de sistemas de independência que são, e que não são, matróides.

Consideremos de novo um grafo bipartido. Um subconjunto X de ligações é um *emparelhamento* se não há duas ligações em X a incidir no mesmo ponto. O conjunto das ligações $\{[a, f], [b, j], [d, i]\}$ do grafo da figura 2 é um emparelhamento. O conjunto $\{[a, f], [b, j], [d, j]\}$ não é emparelhamento. É óbvio que o conjunto dos emparelhamentos forma um sistema de independência (sobre o conjunto das ligações do grafo), mas não é matróide. De facto, não se obtém um emparelhamento se incluirmos no emparelhamento $\{[a, g]\}$ uma das duas ligações do emparelhamento $\{[a, f], [b, g]\}$.

Um conjunto de ligações de um grafo (não necessariamente bipartido) chama-se *floresta* se não inclui *ciclos*.

O conjunto $F = \{[a, d], [c, e], [c, f], [g, h], [h, i], [i, j], [i, m]\}$ é uma floresta do grafo representado no lado esquerdo da figura 3. O conjunto $F \cup \{[h, m]\}$ não é floresta pois inclui o ciclo $\{[h, i], [i, m], [h, m]\}$.

É claro que quando se eliminam ligações numa floresta ainda se obtém uma floresta e, portanto, que o conjunto das florestas é um sistema de independência definido sobre o conjunto das ligações do grafo. Para ver que este sistema de independência é um matróide, considere a floresta F que está representada no lado direito da figura 3, e seja F' uma floresta arbitrária com mais ligações do que F. Como numa floresta não pode haver mais do que $k - 1$ ligações entre pares de k pontos, F' não pode incluir mais do que

- 1 ligação entre pares de pontos do conjunto $C_1 = \{a, d\}$,
- 2 ligações entre pares de pontos do conjunto $C_2 = \{c, e, f\}$, e
- 4 ligações entre pares de pontos do conjunto $C_3 = \{g, h, i, j, m\}$.

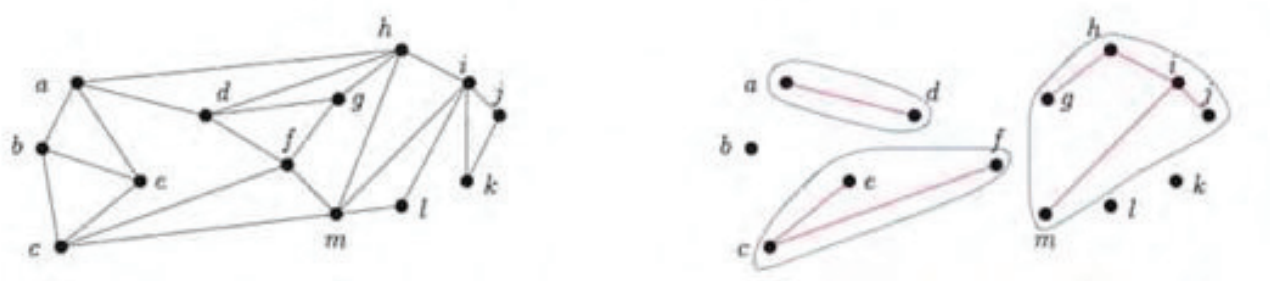

Figura 3: Grafo com 13 pontos (à esquerda) e uma floresta desse grafo com 7 ligações (à direita).

Como |F'| > |F|, vai existir em F' alguma ligação $e = [u,v]$, em que u e v não pertencem simultaneamente a C_1, nem a C_2, nem a C_3. Se juntarmos a F a ligação e, não vamos obter ciclos, i.e., $F \cup \{e\}$ é uma floresta. Espero que o exemplo seja suficientemente ilustrativo para convencer o leitor que o sistema de independência associado às florestas de um grafo é um matróide.

Vamos agora demonstrar o resultado R1. Para demonstrar a implicação $\Leftarrow$ vamos supor que a solução gulosa G não é de custo máximo. Suponhamos então que $G = \{e_1, e_2 \cdots, e_n\}$, com $c_{e_1} \geq c_{e_2} \geq \cdots \geq c_{e_n}$ e que existe um independente (maximal) $O = \{e'_1, e'_2 \cdots, e'_n\}$, em que $c_{e'_1} \geq c_{e'_2} \geq \cdots \geq c_{e'_n}$, com custo $c(O) > c(G)$. Como $c(O) > c(G)$, para algum $1 \leq k \leq n$, tem-se $c_{e'_k} > c_{e_k}$. Seja k o primeiro inteiro em que se verifica a desigualdade anterior. Note que $X = \{e_1, e_2 \cdots, e_{k-1}\}$ ($X = \emptyset$, se $k = 1$) e $Y = \{e_1, e_2 \cdots, e_{k-1}, e_k\}$ são independentes, com $|X| < |Y|$. Logo, algum elemento de Y pode ser inserido em X mantendo a independência. Como todo o elemento de Y tem custo maior do que o custo de e_k, na iteração k, o algoritmo guloso funcionou incorretamente, escolhendo erradamente o elemento e_k! Isto mostra que o algoritmo guloso resolve corretamente o POC associado a um matróide.

Para provar a implicação $\Rightarrow$ vamos supor que o sistema de independência não é um matróide. Sejam então X e Y independentes com $|X| = k < |Y|$, e vamos supor que $X \cup \{y\}$ é *dependente*, qualquer que seja $y \in Y \setminus X$. Definimos custos

$$c_e = \begin{cases} k+2 & \text{se } e \in X \\ k+1 & \text{se } e \in Y \setminus X \\ 0 & \text{em qualquer outro caso} \end{cases}$$

e vamos deixar o algoritmo guloso operar. A solução gulosa G contém X e não inclui elementos de Y. Assim, $c(G) = k(k+2) = k^2 + 2k$. Mas

$$c(Y) \geq (k+1)(k+1) = k^2 + 2k + 1,$$

o que mostra que se o algoritmo guloso encontrar uma solução ótima do POC associado a um sistema de independência $M = (A, S)$, qualquer que sejam os custos atribuídos aos elementos de A, então M é matróide.

Pode concluir-se do resultado R1 que a estratégia gulosa permite, por exemplo, desenhar uma rede viária de menor custo a ligar várias localidades. Basta definir um grafo (*completo*) cujos pontos representam as localidades e associar à ligação entre cada par de pontos u, v o "custo" $c_{[u,v]} = W - w_{[u,v]}$, em que $w_{[u,v]}$ é o custo da construção do troço que liga as localidades u e v, e W um número suficientemente grande para garantir que os custos c sejam positivos. Uma vez que redes com ciclos não podem ter custo mínimo, as redes viárias que interessa considerar são os independentes maximais do matróide das florestas deste grafo. O resultado 1 assegura que o algoritmo guloso determina uma floresta maximal de custo máximo relativamente aos "custos" c. Obviamente, as ligações desta floresta constituem uma rede de menor custo, relativamente aos custos de construção w, a ligar as várias localidades.

Pode também concluir-se do resultado 1 que, provavelmente, não se atinge a maior produtividade possível utilizando a estratégia gulosa para atribuir n tarefas a n operadores. Mais precisamente, suponhamos que se conhecem valores que quantificam os desempenhos dos operadores na execução de cada uma de n tarefas, e que se pretende atribuir a cada operador uma tarefa de forma a maximizar a soma dos valores dos desempenhos. As soluções admissíveis deste POC são os emparelhamentos com n ligações do grafo bipartido completo, em que as classes de bipartição são os conjuntos dos pontos O e T que representam os operadores e as tarefas, respetivamente. Se para todo o par de pontos $u \in O$ e $v \in T$, associarmos o valor do desempenho do operador representado por u na execução da tarefa representada por v, o problema consiste em determinar um emparelhamento, com n ligações, cuja soma dos desempenhos é máximo. Uma vez que o grafo é completo, a solução gulosa é um emparelhamento com n ligações e é portanto admissível. Mas o conjunto dos emparelhamentos do grafo não define um matróide e assim, muito provavelmente, existem soluções mais eficientes do que a solução gulosa para atribuir n tarefas a n operadores.

Ainda no que respeita a emparelhamentos é pertinente observar o seguinte. Um emparelhamento de um grafo bipartido, com classes de bipartição E e D, é um conjunto de ligações que não inclui mais do que uma ligação incidente em cada ponto de E e em cada ponto de D. Facilmente se pode verificar que os conjuntos de ligações, em que não mais do que uma ligação é incidente em cada ponto da mesma classe de bipartição, formam um matróide (sobre o conjunto das ligações do grafo). Assim, os emparelhamentos são os independentes comuns a dois matróides ((i) não mais do que uma ligação

incidente em cada ponto de E e (ii) não mais do que uma ligação incidente em cada ponto de D). Para além do algoritmo guloso que é um método *eficiente* para o POC associado a um matróide, existem também algoritmos *eficientes* para o POC associado à intersecção de dois matróides. É portanto possível identificar, em tempos razoáveis de computação, emparelhamentos de custo máximo e, em particular, distribuir n tarefas a n operadores assegurando a maior produtividade.

Voltemos ao problema da cobertura mínima que, como vimos, não é convenientemente solucionado pela estratégia gulosa. Também não são os algoritmos para o POC na intersecção de dois matróides que vão permitir resolver *eficientemente* o problema, uma vez que não é possível estabelecer para a cobertura mínima uma formulação de POC associado à intersecção de dois matróides. A possibilidade de serem concebidos algoritmos *eficientes* é muito limitada, pois pode provar-se que determinar coberturas mínimas é um *problema difícil*. Um problema é *difícil* se é pelo menos tão difícil quanto um *problema difícil*. Claro que esta definição não teria grande interesse se não existissem problemas comprovadamente *difíceis*. Porém, atualmente é sabido que muitos problemas das mais variadas áreas (lógica, álgebra, grafos, teoria dos números,...) são *problemas difíceis*. Um deles é o *problema da realização*[1] (PR) de expressões booleanas (na forma conjuntiva normal). As ocorrências do PR são coleções de grupos de *literais* associados a variáveis boleanas. Os literais associados a uma variável boleana u são u e $\bar{u}$. Se a variável u toma o valor verdade (falso), os literais u e $\bar{u}$ tomam os valores verdade e falso (falso e verdade), respetivamente.

Por exemplo, a expressão

$$L = \{(x, \bar{y}, z), (x, y, \bar{z}), (\bar{x}, \bar{y}), (\bar{x}, z)\}$$

envolve os pares de *literais* $x, \bar{x}$; $y, \bar{y}$ e $z, \bar{z}$, associados às variáveis boleanas x, y e z, respetivamente. Os grupos de literais $c_1 = (x, \bar{y}, z)$, $c_2 = (x, y, \bar{z})$, $c_3 = (\bar{x}, \bar{y})$ e $c_4 = (\bar{x}, z)$ chamam-se *cláusulas*. Uma afetação de valores verdade-falso sobre as variáveis realiza uma cláusula se, para essa afetação, algum dos literais da cláusula toma o valor verdade. Uma afetação de valores verdade-falso sobre as variáveis realiza a expressão se realiza todas as cláusulas simultaneamente. A afetação $(x \leftarrow V, y \leftarrow F, z \leftarrow F)$, em que V significa verdade e F falso, sobre as variáveis x, y, z, não realiza a expressão L, uma vez que não realiza a cláusula c_4. A afetação $(x \leftarrow V, y \leftarrow F, z \leftarrow V)$ realiza a expressão L.

O PR consiste em decidir, dada uma expressão boleana, se existe uma afetação de valores sobre as variáveis que a realize. Se existir, a expressão diz-se *realizável*.

Vamos converter o PR no problema de encontrar coberturas mínimas. Por outras palavras, para cada expressão boleana (na forma conjuntiva normal), vamos construir um grafo bipartido e mostrar que a expressão é realizável se e só se a cardinalidade das cobertura mínimas do grafo igualar um determinado valor.

Cada variável boleana u vai originar dois pontos no grafo, associados aos literais u e $\bar{u}$. O conjunto dos pontos assim obtidos forma a classe de bipartição E do grafo. Cada cláusula c_i dá origem a um ponto de D, a outra classe de bipartição. Ligamos cada ponto p de E a cada um dos pontos em D que representam as cláusulas que incluem o literal que p representa. Assim, qualquer subconjunto S de pontos E cobre os pontos de D que representam as cláusulas em que figuram os literais representadas pelos pontos de S. Por fim, amplia-se D com um conjunto de pontos A, com número de pontos igual ao número de variáveis e liga-se cada ponto de A com exatamente dois pontos de E, que representam os dois literais associados à mesma variável. Na figura 4 é apresentado o grafo que se obtém com a expressão L, em que os pontos azuis representam os pontos de A.

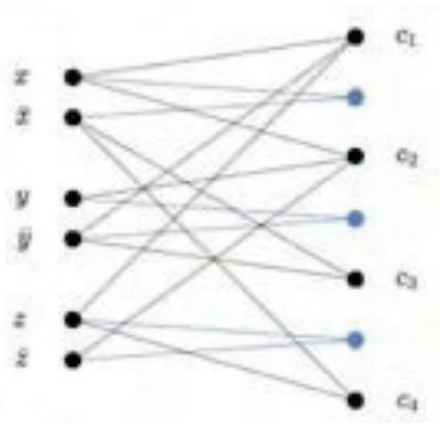

Figura 4: Grafo bipartido correspondente à expressão $L = (c_1, c_2, c_3, c_4)$, com $c_1 = (x, \bar{y}, z)$, $c_2 = (x, y, \bar{z})$, $c_3 = (\bar{x}, \bar{y})$, $c_4 = (\bar{x}, z)$.

[1] Satisfiability problem

Podemos identificar as afetações de valores verdade-falso sobre as variáveis, com os subconjuntos de E que incluem exatamente um dos dois pontos que representam os literais associados à mesma variável. A afetação $(x \leftarrow V, y \leftarrow F, z \leftarrow F)$sobre as variáveis de L corresponde ao conjunto $\{x, \bar{y}, \bar{z}\}$ dos pontos de E do grafo da figura 4. A afetação $(x \leftarrow V, y \leftarrow F, z \leftarrow V)$ corresponde ao conjunto $\{x, \bar{y}, z\}$.

Uma afetação de valores verdade-falso realiza a expressão se e só se o correspondente conjunto de pontos de E é cobertura. A afetação $(x \leftarrow V, y \leftarrow F, z \leftarrow F)$ não realiza a expressão L, uma vez que a cláusula c_4 não inclui um dos literais x, $\bar{y}$ ou $\bar{z}$. Equivalentemente, no grafo da figura 4, o conjunto $\{x, \bar{y}, \bar{z}\}$ não é cobertura (dos pontos de D), pois o ponto que representa a cláusula c_4 não está ligado a qualquer dos pontos desse conjunto. O conjunto de pontos $\{x, \bar{y}, z\}$ do grafo é a cobertura que corresponde à afetação $(x \leftarrow V, y \leftarrow F, z \leftarrow V)$, que realiza L.

Temos pois definida uma correspondência biunívoca entre afetações de valores lógicos sobre as variáveis e subconjuntos de pontos de E, com exatamente um dos dois pontos que representam os literais associados à mesma variável. Uma afetação realiza a expressão se e só se o correspondente conjunto de pontos é cobertura.

Qualquer cobertura (de D) tem de incluir um dos dois pontos que representam os literais associados à mesma variável. Caso contrário, o ponto de A adjacente a esses dois pontos ficaria por cobrir. Assim, por um lado, não há coberturas com cardinalidade menor do que o número de variáveis. Por outro lado, uma cobertura com cardinalidade igual ao número de variáveis, se existir, inclui exatamente um dos dois literais associados a cada variável, e portanto corresponde a uma afetação de valores lógicos que realiza a expressão.

Pode assim concluir-se que a expressão dada é realizável se e só se a cardinalidade de uma cobertura mínima do grafo obtido pelo processo descrito atrás é igual ao número de variáveis boleanas.

Portanto, qualquer algoritmo para o problema da cobertura mínima pode ser utilizado para decidir se uma expressão boleana é realizável. Apenas há que, previamente, executar os procedimentos que definem o grafo associado à expressão dada. Como PR é um *problema difícil*, determinar coberturas mínimas é também um *problema difícil*, o que é interpretado pela maioria dos especialistas da área da complexidade computacional como uma evidência da não existência de algoritmos *eficientes*.

Dado este resultado negativo, onde é que vamos instalar os postos para vigia de fogos florestais? Quais as áreas a selecionar para uma rede de áreas para proteção da biodiversidade? Como escolher os candidatos a guias, em idiomas diferentes, para um museu? A otimização combinatória e, em particular, a teoria dos poliedros combinatórios, são as áreas da matemática nas quais se desenvolvem os fundamentos para responder convenientemente a estas questões[2]. Muitos resultados moderam o pessimismo decorrente da constatação da dificuldade de resolução dos problemas e proporcionam uma narrativa bem mais otimista da localização de postos de vigia de fogos florestais e outras coisas, que se reserva para uma eventual ocasião futura.

SOBRE O AUTOR

Jorge Orestes Cerdeira tem doutoramento em Matemática pela Faculdade de Ciências da Universidade de Lisboa e agregação pelo Instituto Superior de Agronomia da Universidade Técnica de Lisboa, tendo lecionado nesta última instituição até recentemente. Atualmente é professor do Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Os seus interesses científicos incluem otimização combinatória, grafos, macroecologia e biologia da conservação.

[2] Sobre métodos e algoritmos em otimização combinatória o livro de Alexandre Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer, 2003, é uma obra absolutamente notável e, seguramente, a mais completa sobre o assunto.