Modelo entero del problema de supresión de celdas

Modelo entero

A continuación se realiza una descripción formal del problema de supresión de celdas (CSP). En esta sección, se introduce el modelo matemático de programación lineal entera expuesto en detalle en el artículo de Fischetti y Salazar [1].

Los elementos de partida para el planteamiento del problema son los siguientes:

- Colección de celdas que se desea publicar.
- Serie de restricciones lineales que satisfacen las celdas. Éstas vienen fijadas por los marginales, subtotales, tablas enlazadas u otras operaciones entre celdas.
- El intervalo en el que se encuentran los valores publicados ([lb_i,ub_i]).

Matemáticamente estas relaciones se pueden expresar como:

$$\left.\begin{array}{ll} \sum_{i=1}^{n} M_{ij} y_j = b_i & \\ lb_i \leq y_i \leq ub_i & \forall\, i \in \{1, \ldots n\} \end{array}\right\} \qquad (1)$$

donde,

i es el número de variables (celdas)
j es el número de ecuaciones del problema
M_{ij} es la matriz que expresa las relaciones aritméticas de las variables
y_j corresponde al valor de la celda i en la publicación,
b_i son los marginales
lb_i, ub_i define el intervalo conocido en el que se encuentra el valor de la celda.

Para la resolución del problema CSP se introduce una variable de decisión binaria por cada celda publicada; es decir, $\boldsymbol{x_i} \in \{0,1\}$ para todo ***i*** desde 1 hasta el número de celdas. Si en la solución del problema CSP su valor es cero, la celda será publicada; en caso contrario, la celda debe ser suprimida.

Debe definirse el ***umbral de confidencialidad estadística***[1] que determina qué cedas deben suprimirse (supresiones primarias) y debe determinarse si el valor cero será ocultado o no. Así mismo, se establecen los niveles de protección por cada celda. Se trata del valor máximo deducible[2] de la celda, el mínimo y la diferencia entre ambos (estos valores son denominados *lpl, upl, spl*; y provienen de los términos ingleses lower protection level, upper protection level y slide protection level).

Para representar el problema CSP asociado a la búsqueda de las supresiones secundarias que minimizan la pérdida de información manteniendo la confidencialidad existen esencialmente dos modelos de programación entera.

El primer modelo, llamado modelo clásico, propuesto por Kelly (1990), expresa el problema CSP como un problema de programación lineal entera (ver [3] para una estimación del tamaño del problema según el tamaño de las tablas, el número de celdas marginales y el número de supresiones

[1] El umbral de confidencialidad es un número entero que marca el límite por debajo del cual una celda, que represente un número de declarantes, se considera sensible y no se debe mostrar en la publicación.

[2] Si no se introducen determinadas supresiones secundarias, a partir de los marginales, por ejemplo, publicados es posible desvelar datos que hayan sido ocultados por cuestiones de confidencialidad.

primarias). Aun tratándose de un problema de programación lineal relajado[3], implica un número muy alto de variables auxiliares y de restricciones asociadas que hacen en la práctica inviable la resolución del problema, en un tiempo razonable, utilizando los optimizadores lineales con los que se realizaron pruebas: GLPK, CPLEX, XPress...

El segundo planteamiento, propuesto por Fischetti y Salazar detalladamente en [1], lleva al siguiente problema de programación entera:

$$\min \sum_{i=1}^{n} w_i \; x_i \qquad (2)$$

donde los elementos de esta función objetivo son:

- x_i son las variables de decisión que definen si la celda i se oculta o se publica.
- w_i establece el coste en información de ocultar la celda i.

Las restricciones que se imponen a la anterior función objetivo son los niveles de protección de cada celda confidencial (supresión primaria). En el caso de un único atacante externo estos niveles (lpl, upl, spl), pueden establecerse como un problema lineal para cada celda.

Por ejemplo, la condición del límite superior (upl), para cada celda $\mathbf{i_k}$, se corresponde con el siguiente problema lineal:

$$\left.\begin{array}{l} \overline{y}_{i_k} = \max y_{i_k} \,, \\ \text{Sujeto a} \\ My = b, \\ y_i \leq \quad a_i + UB_i x_i \qquad \forall i \in \{1, \ldots n\} \\ -y_i \leq -a_i + LB_i x_i \qquad \forall i \in \{1, \ldots n\} \end{array}\right\} \qquad (3)$$

Si el valor del óptimo, $\overline{y}_{i_k}$, es superior al valor de la celda $\mathbf{i_k}$ + upl_{i_k}, la celda confidencial se encuentra protegida para las supresiones, $\mathbf{x_i}$, propuestas. En caso contrario, el conjunto de supresiones no cumple los niveles de protección superior requeridos para la celda $\mathbf{i_k}$. Si no se cumplen los niveles de protección se deberá añadir, al problema inicial, una restricción adicional que garantice la confidencialidad exigida. Estas restricciones adicionales se denominan restricciones de capacidad (***capacity constraints***) y posteriormente, en el apartado relativo al ***Algoritmo empleado en la resolución***, se detallará cómo se calculan.

El tipo de problema definido en (3) es conocido como ***subproblema del atacante***. El número de estos problemas crece en el caso de considerarse múltiples atacantes externos.

Los problemas que representan los límites inferior (lpl) y de tamaño de intervalo (spl), para la celda $\mathbf{i_k}$, son análogos al expuesto anteriormente, ver [1].

El problema completo CSP consiste, por lo tanto, en optimizar la función objetivo (2), sujeta a las restricciones definidas en (1) y a las restricciones adicionales derivadas de los niveles de protección definidos por los subproblemas del atacante (3) para cada celda confidencial.

En definitiva, el modelo entero que representa al problema CSP es un modelo binivel entero, con un número exponencial de restricciones lineales establecidas para garantizar los niveles de protección requeridos.

[3] Relajar el problema consiste en trabajar con números reales en lugar de con números enteros. Por lo tanto; para nuestro caso, las variables binarias xi $\in$ {0, 1} pasan a cumplir la siguiente condición: 0 $\leq$ xi $\leq$ 1.

Algoritmo empleado en la resolución

El proceso de resolución que se expone en este apartado, consiste en obtener una solución que optimice la función objetivo definida en (2); es decir, encontrar aquel conjunto de celdas a suprimir que garanticen la mínima pérdida de información. Posteriormente, se intenta vulnerar la confidencialidad de la solución generada y se comprueban cuáles son las restricciones de confidencialidad que un atacante externo puede encontrar para el máximo, el mínimo y el intervalo en el que se encuentra una celda. Si dichos valores cumplen los niveles de protección exigidos, para todas y cada una de las celdas consideradas supresiones primarias, el problema ha finalizado; en caso contrario, las restricciones que no satisfacen los niveles de protección sirven para definir nuevas condiciones que se deben añadir al problema y repetir el proceso.

La solución de problemas binivel se suele obtener empleando un esquema iterativo en el que se van añadiendo planos de corte. Este procedimiento se denomina método de ramificación y poda (Branch-and-Cut) ver [4]. Aplicando este procedimiento de ramificación, se llega a conseguir la solución del problema binivel en un tiempo polinómico.

El algoritmo se inicia con un problema principal (***problema master***) definido por la siguiente función objetivo:

$$\min\left\{\sum_{i=1}^{n} w_i x_i :\ x_{i_1} = \cdots = x_{i_p} = 1, x \in [\![0,1]\!]^n\right\} \qquad (4)$$

donde:

- x_i son las variables de decisión que definen si la celda i se oculta o se publica.
- w_i establece el coste en información de ocultar la celda i.

A la que se imponen las siguientes condiciones:

- Las celdas consideradas supresiones primarias; es decir, su valor está por debajo del umbral elegido, se suprimen de la publicación. Esto se expresa matemáticamente: $x_{i_1}, \dots, x_{i_p} = 1$
- Por cada ecuación en la que participe una celda de supresión primaria, se debe suprimir además al menos otra celda (supresión secundaria).

Proceso de ramificación

La resolución del problema inicial sobre los enteros (en nuestro caso variable binaria 0,1) tiene una mayor complejidad que su resolución sobre los reales (problema relajado). Por este motivo; en primer lugar, se encuentra la solución real del problema que cumple todas las restricciones de confidencialidad y; posteriormente, se va ramificando la solución real.

El proceso de ramificación (fase Branch) consiste en asignar un valor 0 ó 1 a cada elemento de la solución real descartando ramas de forma inteligente. Todas las posibles ramas representan las diferentes combinaciones de valores enteros (binarios en nuestro caso) cercanos a la solución real.

Proceso de poda

A continuación, se detallan los pasos a seguir para añadir restricciones al problema (fase Cut). Este proceso se repite en cada una de las ramas incluso en la rama principal del árbol constituida por el problema inicial.

En primer lugar, se obtiene una solución óptima, ***x****, del problema inicial y se comprueba si las supresiones propuestas, ***x****, cumplen los requisitos de confidencialidad definidos:

- En caso afirmativo, el problema está resuelto y tenemos una solución con pérdida mínima de información que cumple los criterios de seguridad en la publicación.
- En caso contrario, tenemos un problema de separación asociado a las restricciones de capacidad. Este problema se resuelve siguiendo el algoritmo que se expone a continuación. Por cada supresión primaria, se realizan estos pasos:
  1. Se soluciona el subproblema del atacante, para el nivel de protección superior (upl) similar a (3), y se chequea si se cumple la restricción de confidencialidad superior para la celda:
     a. En caso afirmativo, continuamos con el siguiente paso.
     b. En caso negativo, generamos una restricción de capacidad, a partir de la condición violada por la solución propuesta ***x****. Esta restricción se obtiene a partir de la solución del *problema dual al subproblema del atacante* (3) que incumple la protección exigida.

        El dual del subproblema del atacante se formula en el siguiente problema lineal:

$$\left.\begin{array}{l} \bar{y}_{i_k} = \min \Upsilon^t\, b + \sum_{i=1}^{n}\left(\alpha_i(a_i + UB_i x_i) - \beta_i(a_i - LB_i x_i)\right), \\ \text{Sujeto a} \\ \alpha^t - \beta^t + \Upsilon^t M = e_{i_k}^t \\ \alpha \geq 0, \beta \geq 0, \Upsilon \text{ sin restricción en el signo} \end{array}\right\} \quad (5)$$

        La nueva restricción de capacidad generada tiene la forma:

$$\sum_{i=1}^{n}(\propto_i UB_i + \beta_i LB_i)x_i \geq UPL_k))$$

        Para todos los puntos extremos (***α,β,γ***) que satisfacen el problema dual (5) al subproblema del atacante.
  2. Se soluciona el subproblema del atacante para el nivel de protección inferior (lpl) y se chequea si cumple todas las restricciones de confidencialidad. Se procede del mismo modo que en el anterior apartado:
     a. En caso afirmativo, continuamos con el siguiente paso.
     b. En caso negativo, generamos una restricción de capacidad violada por la solución propuesta ***x**** de la forma:

$$\sum_{i=1}^{n}(\propto_i UB_i + \beta_i LB_i)x_i \geq LPL_k))$$

        Para todos los puntos extremos (***α,β,γ***) que satisfacen el problema dual al subproblema del atacante asociado a la restricción de confidencialidad inferior (lpl).
  3. Empleando las soluciones de los problemas del atacante anteriores se chequea el nivel de protección del intervalo de confidencialidad (spl):
     a. En caso afirmativo, continuamos con el siguiente paso.
     b. En caso negativo, generamos una restricción de capacidad violada por la solución propuesta *x**. Esta restricción tiene la forma:

$$\sum_{i=1}^{n}((\propto_i+\propto_i')UB_i + (\beta_i + \beta_i')LB_i)x_i \geq SPL_k))$$

        Para todos los puntos extremos (***α, α', β, β', γ, γ'***) que satisfacen los problemas duales a los subproblemas del atacante anteriores.

Después de haber recorrido todas las celdas primarias, se comprueba si se han añadido nuevas restricciones de capacidad:

a. En caso negativo, la solución propuesta **x***, es la supresión que buscamos y cumple todas las restricciones de confidencialidad.
b. En caso de haberse generado alguna nueva restricción debemos realizar una nueva optimización del problema inicial. De este modo se generará una nueva solución **x**** a partir de la cual se repetirá el proceso.

Optimización del algoritmo

El algoritmo incluye mejoras basadas en el fortalecimiento de las restricciones del problema lineal. Principalmente, se trata de desigualdades de cobertura, eliminación de desigualdades puente, etc. Algunas de estas mejoras son directamente implementadas por el optimizador lineal empleado y otras han sido añadidas en el desarrollo (para una exposición detallada de estas técnicas consultar el artículo [1]).

Para aumentar la velocidad de resolución del problema se realiza un preproceso del mismo, en el que se eliminan del problema celdas primarias que obtienen protección automáticamente del resto de celdas primarias. De esta manera, se consigue reducir la dimensión del problema.

Uso de soluciones heurísticas

La velocidad de convergencia del algoritmo iterativo de branch-and-cut puede incrementarse si se halla antes una solución heurística del problema. Esta solución, generalmente distinta de la óptima, se emplea para desestimar ramas del árbol de decisión.

El heurístico implementado se basa en las ideas de Nelly y Robertson [2], [3], tal y como se describe en [1].

El procedimiento se ejecuta por pasos:

- Se comienza con un conjunto de supresiones inicial que coincide con las supresiones primarias.
- Para todas las celdas con supresiones primarias:
  - Se encuentra un conjunto de supresiones que garantizan los niveles de protección para cada celda; upl, lpl y spl (resolviendo los problemas del atacante correspondientes).
  - Las nuevas supresiones se añaden al conjunto inicial. Por lo tanto, para cada nueva celda primaria analizada lo normal es que se haya ampliado el conjunto de supresiones secundarias.
- Se realiza un procedimiento de limpieza de las supresiones halladas para eliminar redundancias dado que algunas supresiones pueden sobreproteger celdas que ya se encontraban protegidas.
  De forma iterativa se intenta eliminar las supresiones redundantes de mayor peso.

Referencias

[1] Fischetti y Salazar (2001): Solving the cell suppression problem on tabular data with linear constraints. Management Science.
[2] Robertson D.A. (1995) Cell Suppression at Statistics Canada. Proc. Second International Conference in Statistical Confidentiality
[3] Kelly J.P., Golden L.A. (1992) Cell Suppression Disclosure protection for sensitive tabular data.
[4] Nemhauser, G. L., L. A. Wolsey. 1988. Integer and combinatorial optimization, John Wiley & Sons.