Grado en Ingeniería Informática
Diseño y Desarrollo de Sistemas de Información
Resolución del examen de contenidos teóricos, Febrero de 2016

Ejercicio 1

Las metaheurísticas son una familia de algoritmos aproximados de propósito general usados para resolver instancias complejas de una gran variedad de problemas de optimización. El término *metaheurística* apareció por primera vez en un artículo de Fred Glover en 1986. Se desea diseñar un sistema para tener un registro de éstas y de los problemas e instancias que resuelven, de modo que cumpla las siguientes restricciones:

a) Un problema se identifica por un nombre y tiene una descripción, y se compone de instancias que se identifican por el nombre del problema y un identificador.

b) Una metaheurística puede resolver diferentes instancias, y una instancia puede ser resuelta por diferentes metaheurísticas.

c) Las metaheurísticas se identifican por su nombre y se caracterizan por una descripción. Éstas son de uno de dos tipos diferentes: de trayectoria y de población.

d) Las metaheurísticas de población usan dos operadores: un operador de mutación y un operador de cruce. Cada operador puede ser usado en varias metaheurísticas de población.

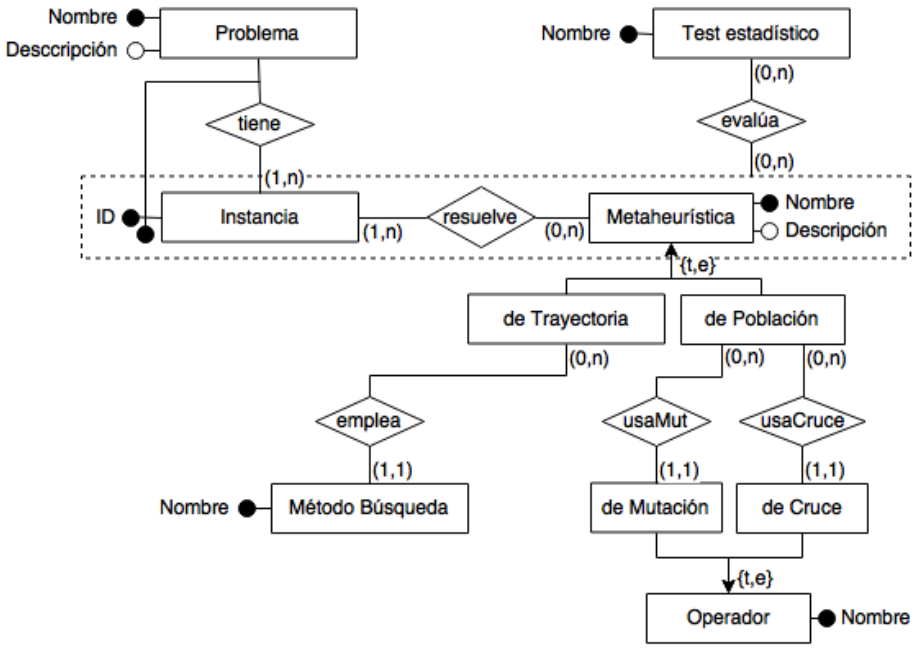
e) Los operadores se identifican por un nombre y tienen que ser de uno de los dos tipos exclusivamente.

f) Las metaheurísticas de trayectoria emplean un método de búsqueda, aunque el mismo método de búsqueda puede ser usado en varias metaheurísticas de trayectoria. Cada método de búsqueda se identifica por un nombre.

g) Se aconseja realizar varios tests estadísticos para evaluar y comparar el rendimiento de las metaheurísticas cuando resuelven instancias de un problema, aunque el mismo test puede usarse para evaluar el rendimiento de varias heurísticas cuando resuelve instancias de uno o varios problemas. Estos tests se identifican por un nombre.

Diseña un diagrama entidad-relación que recoja todas estras restricciones y permita que el sistema realice todas las funcionalidades especificadas.

Resolución


Nombre
Descripción
Problema
Nombre
Test estadístico
(0,n)
tiene
evalúa
(0,n)
(1,n)
ID
Instancia
(1,n)
resuelve
(0,n)
Metaheurística
Nombre
Descripción
{t,e}
de Trayectoria
de Población
(0,n)
(0,n)
(0,n)
emplea
usaMut
usaCruce
(1,1)
(1,1)
(1,1)
Nombre
Método Búsqueda
de Mutación
de Cruce
{t,e}
Operador
Nombre

Ejercicio 2

Propón un diagrama E/R que sea consistente con el siguiente conjunto de tablas (esquema relacional) en el que los atributos que forman parte de la llave primaria aparecen en negrita y las llaves externas y restricciones adicionales (*NOT NULL*) se especifican explícitamente para cada tabla:

a) Producto (**idprod**)

b) Caja (**idprod**, nbotellas), con *idprod* llave externa a la tabla *producto*

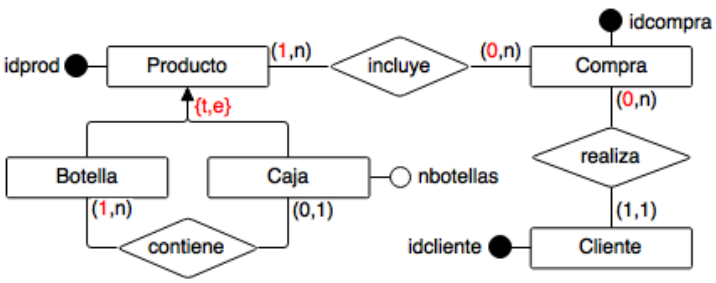
c) Botella (**idprod**, idcaja), con *idprod* llave externa a la tabla *producto* y *idcaja* llave externa al atributo *idprod* de la tabla *caja*

d) Cliente (**idcliente**)

e) Compra (**idcompra**, idcliente), con *idcliente* llave externa a la tabla *cliente* y una restricción Not Null sobre dicho atributo

f) Incluye (**idcompra**, **idprod**), con *idcompra* llave externa a la tabla *compra* y *idprod* llave externa a la tabla *producto*.

Resolución



Las participaciones y exclusividades (solapamientos) que aparecen en rojo en el diagrama no son necesariamente esas. Es decir, que distintas combinaciones de ellas darían como resultado el mismo esquema relacional.

Ejercicio 3

Sea R una relación de esquema {A, B, C, D, E, F} que verifica el conjunto F de dependencias funcionales {A→F, AE→B, E→D, AB→F, F→E}. Determinar si ABC, ACE y ACF son claves candidatas. Determinar, asimismo, si hay alguna otra clave candidata de la relación según el conjunto de dependencias funcionales.

Resolución

Se proponen dos formas para comprobar la validez de las claves propuestas y la búsqueda de claves adicionales, si las hubiera.

Resolución mediante las propiedades de atributos en las claves

Basta con comprobar las características de los atributos que conforman las claves propuestas y que estas, en sí mismas, no sean *superclaves* (una ampliación de una clave, es decir, que incluyen a una clave en su interior).

Respecto de los atributos se ha de cumplir:

- Los atributos independientes (según el conjunto de dependencias funcionales) deben formar parte de todas las claves candidatas y, por lo tanto, de las propuestas. El único atributo independiente es C y forma parte de las tres claves propuestas.
- Los atributos que sólo aparecen a la izquierda de cualquier dependencia funcional (atributos determinantes y no determinados, según el conjunto de dependencias funcionales) deben formar parte de todas las claves candidatas y, por lo tanto, de las propuestas. El único atributo de este tipo es A y forma parte de las tres claves propuestas.
- Los atributos que sólo aparecen a la derecha de cualquier dependencia funcional (atributos determinados y no determinantes, según el conjunto de dependencias funcionales) no pueden formar parte de ninguna clave candidata y, por lo tanto, de ninguna de las propuestas. El único atributos de este tipo es D y no forma parte de ninguna de las tres claves propuestas.
- Los atributos que aparecen a la izquierda en alguna dependencia funcional y a la derecha en alguna otra (atributos determinantes y determinados, según el conjunto de dependencias funcionales) pueden formar parte de cualquier clave candidata y, por lo tanto, de alguna de las propuestas. Los atributos de este tipo son B, E y F, que forman parte de una de las tres claves propuestas.

Respecto de las superclaves, se ha de cumplir que cada clave de las propuestas no incluya en su interior un subconjunto que sea clave en sí mismo. El procedimiento más sencillo para comprobarlo es el de búsqueda de *atributos raros* (un atributo es *raro* respecto de un conjunto de atributos si es determinado por este, es decir, que se puede *alcanzar* el atributo a partir de los demás mediante las dependencias funcionales del conjunto). Probemos si cada atributo es raro con respecto a los demás:

- Empezamos por la primera de las tres claves propuestas ABC:

- A sería raro con respecto a BC si A pudiera alcanzarse desde BC; para probar esto, sólo hay que calcular el cierre transitivo de atributos de BC, BC^+={B, C}, el cual no incluye a A, de lo que podemos deducir que A no es alcanzable sólo con los atributos B y C y, por tanto, **A no es raro respecto de BC**.
- B sería raro con respecto a AC si B pudiera alcanzarse desde AC; para probar esto, sólo hay que calcular el cierre transitivo de atributos de AC, AC^+={A, B, C, D, E, F}, que incluye a B, de lo que podemos deducir que B es alcanzable con los atributos A y C y, por tanto, **B es raro respecto de AC**. La consecuencia de esto es que AC→B y, por tanto, ABC no puede ser una clave. También, del cálculo de AC^+ deducimos que AC es una clave porque nos permite obtener todos los atributos de la relación.
- Puesto que B debe ser eliminado de la combinación con A y C, queda probar si:
  - A no es raro con respecto a C puesto que no hay dependencias que incluyan a C y, por tanto, el único atributo alcanzable desde C es el propio C.
  - C no es raro con respecto a A puesto que no hay dependencias que incluyan a C y, por tanto, no se puede alcanzar a ningún atributo sólo con C.

El resultado de este proceso elimina ABC como clave candidata y añade a AC como clave.

- Dado que hemos determinado en el paso anterior que AC es una clave, eso haría que ACE sea una extensión de la clave (porque incluye a AC) y, por tanto ACE no puede ser clave.
- Dado que hemos determinado en un paso anterior que AC es una clave, eso haría que ACF sea una extensión de la clave (porque incluye a AC) y, por tanto ACE no puede ser clave.

De modo que AC es una clave, por lo que ABC, ACE y ACF no pueden serlo (al ser extensiones) y son las únicas que podrían porque son las únicas que incluyen a todos los atributos independientes, determinantes no determinados y determinantes determinados.

La conclusión es que **las claves propuestas no son correctas y la única clave existente es AC**.

<u>Resolución mediante el cálculo de las claves de la relación mediante el algoritmo correspondiente</u>

En primer lugar, sería recomendable establecer los tipos de atributos en función de si aparecen a la izquierda en todas las dependencias funcionales en las que aparece, a la derecha en todas las dependencias funcionales en las que aparece, a la izquierda de alguna y a la derecha en otra o en ninguna dependencia funcional:

- atributos independientes (no aparecen en ninguna dependencia): {C}
- atributos equivalentes: no existen equivalencias explícitas entre parejas de atributos
- atributos que aparecen a la izquierda en todas las dependencias en las que aparecen: {A}
- atributos que aparecen a la izquierda de alguna dependencia funcional y a la derecha de otra: {B, E, F}
- atributos que aparecen a la derecha en todas las dependencias en las que aparecen: {D}

Establecidos los tipos, podemos aplicar el algoritmo de extracción de claves candidatas:

1º Eliminación de atributos independientes:

La relación sin independientes queda $R_{SI} = R - \{C\} = \{A, B, D, E, F\}$

2º Eliminación de equivalencias:

Al no existir equivalencias entre parejas de atributos $R_{SIE} = R_{SI} = \{A, B, D, E, F\}$ y $F_{SIE} = F$.

3º Procesamiento de atributos que aparecen sólo a la izquierda:

$K_p = \{A\} = A$

Habiendo un candidato, tenemos que probar si es una clave:

$A^+ = \{A, B, D, E, F\} = R_{SIE}$, $CK_{SIE} = \{A\}$ y, al haber obtenido una clave en el paso 3, no procede pasar al paso 4 puesto que implicaría combinar otros atributos con A, que ya es clave.

4º Procesamiento de atributos que aparecen a la izquierda y a la derecha:

No procede.

5º Incorporación de atributos independientes:

Puesto que existían atributos independientes en R, este paso modifica las claves obtenidas para la relación R_{SI}, añadiendo los atributos independientes a todas las claves por lo que CK' = {AC}.

6º Incorporación de atributos equivalentes:

Al no haber equivalencias entre parejas de atributos en F, este paso no tiene que generar versiones de claves, por lo que, CK = {AC}

El hecho de haber aplicado el algoritmo de extracción de claves candidatas, garantiza que se ha encontrado la única que hay: AC.

Ejercicio 4

Sea R una relación de esquema {A, B, C, D, E}, con un conjunto CK de claves candidatas {AB, BC} y que verifica un conjunto F de dependencias funcionales {B→D, AD→C, CD→A, BD→E}. Comprobar si la relación está en Forma Normal de Boyce y Codd (sin comprobar si está en segunda forma normal o en tercera forma normal) y si no lo está, realizar una descomposición hasta que todas las relaciones de la descomposición estén en dicha forma normal.

Resolución

La relación R estaría en Forma Normal de Boyce y Codd si, y sólo si, todas las dependencias tienen a su izquierda una clave candidata. Según esta norma, podemos decir que la relación no está en BCNF porque B→D, AD→C, CD→A y BD→E forman parte de F y ni B ni AD ni CD ni BD son claves candidatas de la relación.

Para conseguir una normalización, debemos aplicar el Algoritmo de Heath sobre una de las dependencias que no cumplen la forma normal. Sin embargo, se sabe que si se considera la importancia de las dependencias según el atributo que figura a su derecha, se reduce el riesgo de la pérdida de dependencias en una descomposición. La importancia de las dependencias para la normalización se establece, de menor a mayor, en dependencias que tienen a la derecha un atributo determinado y no determinante (BD→E), dependencias que tienen a la derecha un atributo determinado y determinante que no forma parte de ninguna clave (B→D) y dependencias que tienen a la derecha un atributo determinado y determinante que forma parte de alguna clave (AD→C, CD→A). Escogiendo entre las dependencias "menos importantes para la normalización" minimizamos riesgos, así que normalizamos por BD→E:

R_1 = {B, D, E}, F_1 = {BD→E, B→D}, CK_1 = {B}, R_1 no está en BCNF porque todas sus dependencias tienen una clave candidata a la izquierda. Si volvemos a aplicar el algoritmo sobre esta relación y este conjunto de dependencias volvemos a obtener la misma relación y así, recursivamente. El problema se debe a las dos dependencias y a que BD→E es una dependencia a la que se puede aplicar la regla de pseudotransitividad con B→D para transformarla en B→E y deducir que el conjunto F de partida nos haría caer en un proceso de recursión infinita en la normalización.

R_2 = {A, B, C, D}, F_2 = {AD→C, CD→A, B→D}, CK_2 = {AB, BC}, **R_2 no está en BCNF** porque AD→C, CD→A y B→D forman parte de F_2 pero ni AD ni CD ni B son claves candidatas de R_2.

Es necesario normalizar la relación R_2 mediante la aplicación del Teorema de Heath. Sin embargo, no hay dependencias en F_2 que tengan un atributo determinante y no determinado a la derecha. El orden de importancia de las dependencias de F_2 para la normalización, de menor a mayor, sería B→D, AD→C y CD→A.

Escogiendo la dependencia "menos importante para la normalización" minimizamos riesgos, así que normalizamos por B→D:

$R_{2,1}$ = {B, D}, F_2 = {B→D}, CK_2 = {B}, **R_2 está en BCNF** porque todas sus dependencias tienen a la izquierda una llave candidata.

$R_{2,2}$ = {A, B, C}, F_2 = Ø, CK_2 = {AB, BC}, **R_2 está en BCNF** porque todas sus dependencias tienen a la izquierda una llave candidata.

El resultado de nuestra normalización es el conjunto de relaciones:

$$\{(\{B, E\}, r_1), (\{B, D\}, r_{2,1}), (\{A, B, C\}, r_{2,2})\}$$

Cálculo del recubrimiento minimal F' para la simplificación del conjunto de dependencias funcionales (opcional)

El algoritmo de cálculo del recubrimiento minimal de dependencias funcionales F' nos permite simplificar el conjunto F a la hora del proceso de normalización e, incluso, del de cálculo de llaves candidatas.

El algoritmo consta de tres pasos, que se detallan a continuación.

El **primer paso** construye el conjunto $\mathbf{F^{(1)}}$ en el que se simplifican las partes derechas de las dependencias de F aplicando la regla de descomposición a cada una de las dependencias que tengan varios atributos en su parte derecha. Dado que ninguna dependencia de F tiene una parte derecha compuesta, $F^{(1)}$=F.

El **segundo paso** construye el conjunto $\mathbf{F^{(2)}}$ en el que se simplifican las partes izquierdas de las dependencias de F buscando la existencia de dependencias entre atributos de la parte izquierda (lo que se denominan *atributos extraños* o *raros*) de modo que se pueda aplicar la regla de pseudo-transitividad para reducir el número de atributos en la parte izquierda.

Análizamos la dependencia AD→C en busca de atributos raros, de modo que debemos examinar si A es raro con respecto a D y viceversa.

Se dice que A es raro con respecto a D si, y sólo si, D→A pertenece al conjunto F o se puede deducir de él. Como sabemos, la obtención de esta dependencia puede ser bastante complicada. Pero podemos saber si se puede deducir de F si, y sólo si, se cumple que $A \in D^+$. Como podemos comprobar $D^+ = \{D\}$ y $A \notin D^+$, por lo que A no es raro con respecto a D y no podemos aplicar la regla de pseudo-transitividad para eliminar A de este conjunto de atributos.

Calculamos si D es raro con respecto A pero como podemos comprobar $A^+ = \{A\}$ y $D \notin A^+$, por lo que D no es raro con respecto a A y no podemos aplicar la regla de pseudo-transitividad para eliminar D de este conjunto de atributos.

Procedemos de la misma manera con la dependencia CD→A, para examinar si C es raro con respecto a D o viceversa. Como $C^+ = \{C\}$ y $D \notin C^+$, D no es raro con respecto a C. Como $D^+ = \{D\}$ y $C \notin D^+$, C no es raro con respecto a D. En cualquiera de los casos, no podemos aplicar la regla de pseudo-transitividad sobre esta dependencia al no haber atributos raros en ella.

Procedemos igualmente con la dependencia BD→E, para examinar si B es raro con respecto a D o viceversa. Como $B^+ = \{B,D,E\}$ y $D \in B^+$, D es raro con respecto a B y, por tanto, podemos aplicar la regla de pseudo-transitividad sobre la dependencia BD→E y la dependencia B→D, que acabamos de deducir del cálculo de B^+ (y, además, está en el propio F), dejando la dependencia del conjunto como

BB→E o, lo que es lo mismo, B→E. Dado que hemos eliminado D del conjunto de atributos de la izquierda, ya no podemos seguir buscando atributos raros porque sólo queda uno.

En resumen, nos queda un $F^{(2)}$={B→D, AD→C, CD→A, B→E}.

El **tercer paso** construye el conjunto $\mathbf{F^{(3)}}$, en el que se simplifican dependencias completas, eliminándolas por ser redundantes (deducibles por combinación de otras). Para ello, procesamos cada dependencia eliminándola del conjunto considerado y tratando de deducirla de nuevo de las restantes. Como hemos visto, tratar de deducir una dependencia puede llegar a ser tedioso, pero basta con calcular el cierre de atributos de la parte izquierda de la dependencia con respecto a las dependencias restantes (menos la eliminada). Si la parte derecha de la dependencia está incluida en el cierre, la dependencia es deducible del conjunto de las restantes, incluso después de haberla eliminado, por lo que es redundante y puede eliminarse de $F^{(3)}$ definitivamente.

Empezamos con la evaluación de la posible redundancia de B→D. Evaluamos el cierre B^+ con respecto al conjunto de dependencias restante {AD→C, CD→A, B→E} y obtenemos B^+={B, E}. Dado que D ∉ B^+, la dependencia B→D no puede deducirse de las demás y no es redundante.

Procedemos con la evaluación de la posible redundancia de AD→C. Después de calcular el cierre AD^+ con respecto al conjunto de depentencias restante {B→D, CD→A, B→E}, observamos que C ∉ AD^+={A, D} por lo que la dependencia AD→C no puede deducirse de las demás y no es redundante.

Para ver si CD→A es redundante, basta con calcular CD^+ con respecto al conjunto de dependencias restante {B→D, AD→C, B→E}. Como podemos observar, A ∉ CD^+={C, D} por lo que la dependencia CD→A no puede deducirse de las demás y no es redundante.

Por último, para ver si B→E es redundante, basta con calcular B^+ con respecto al conjunto de dependencias restante {B→D, AD→C, CD→A}. Como podemos observar, E ∉ B^+={B, D} por lo que la dependencia B→E no puede deducirse de las demás y no es redundante.

Después de evaluar cada dependencia del conjunto $F^{(2)}$, y decidir que ninguna es redundante, nos queda un conjunto $F^{(3)}=F^{(2)}$={B→D, AD→C, CD→A, B→E}.

De modo que el recubrimiento minimal de F es F'=$F^{(3)}$={B→D, AD→C, CD→A, B→E}

Como se puede observar, la dependencia funcional que nos ocasionaba un problema con la normalización de la relación R (BD→E) ha desaparecido, al igual que la recursión que aparecía en el algoritmo.