Técnicas para el Manejo de CSPs no Binarios

Miguel A. Salido

Dpto. Ciencias de la Computación e Inteligencia Artificial
Escuela Politécnica Superior
Universidad de Alicante, Spain
{masalido}@dccia.ua.es

Resumen

Hoy en día muchos problemas de la vida real se pueden modelar como problemas de satisfacción de restricciones (CSPs) no binarias (o $n-$arias). Por ejemplo en áreas tales como inteligencia artificial, investigación operativa, bases de datos y sistemas expertos, la importancia de los CSPs no binarios se está incrementando paulatinamente. Sin embargo la mayoría de los investigadores centran su atención en los CSPs binarios, tal vez por el hecho de que un CSP no binario se puede transformar en uno binario. En este artículo se presentan las distintas técnicas para transformar los CSPs no binarios en binarios, para el caso de CSPs discretos, y en ternarios para el caso de CSPs continuos. Además se presenta una propuesta de manejo de CSPs no binarios y continuos. Utilizando esta última técnica resolveremos un problema de diagnosis basado en el análisis ROC. Este problema se modela como un CSP no binario sobre dominios continuos.

Palabras clave: Problemas de Satisfacción de Restricciones, CSPs no binarios.

1. Introducción

Hoy en día, muchos problemas de la vida real pueden modelarse como problemas de satisfacción de restricciones (CSPs) y resolverse usando técnicas de programación de restricciones. Esto incluye problemas de áreas tales como inteligencia artificial, investigación operativa, bases de datos, sistemas expertos, etc. Algunos ejemplos son scheduling, planificación, razonamiento temporal, diseño en la ingeniería, problemas de empaquetamiento, criptografía, toma de decisiones, etc. Un gran número de estos problemas se modelan de una manera natural mediante el uso de restricciones no binarias. Las restricciones no binarias son aquellas que constan de cualquier número de variables. El manejo de restricciones no binarias es un problema NP-completo [15].

El interés actual por el desarrollo e investigación en técnicas de satisfacción de restricciones está suficientemente contrastado en los diferentes foros científicos, bibliografía relevante, aplicaciones destacadas, etc. Particularmente, en la literatura de satisfacción de restricciones, la necesidad de manejar restricciones no binarias ha empezado recientemente a ser reconocido. La mayoría de los investigadores que trabajan en problemas de satisfacción de restricciones centran su atención en problemas binarios y discretos por diversos motivos:

- La razón básica de tratar con restricciones binarias es por la simplicidad que supone comparado con las no binarias y también por el hecho de que todo problema de satisfacción de restricciones no binarias puede transformarse en uno binario equivalente [21, 2].
- Además se trabaja principalmente con do-

minios discretos por la reducción de complejidad que supone trabajar con un conjunto finito de valores para cada una de las variables del problema. De esta manera se puede generar el árbol de búsqueda en un problema dado, y llevar a cabo el estudio de la consistencia mediante la búsqueda sistemática a través de la exploración de dicho árbol.

En el caso de problemas de satisfacción de restricciones con dominios continuos como los CSPs numéricos, muchos investigadores también llevan a cabo una transformación del problema en el cual las restricciones $n-$arias se transforman en restricciones ternarias [27, 14].

Tanto en CSP discretos como continuos, el proceso de transformación puede resultar poco eficiente debido a su alto coste computacional. Por lo tanto, a menudo resulta más adecuado manejar las restricciones no binarias de forma directa, de manera que los problemas mantengan sus formulaciones originales.

En general, las dos principales cuestiones con la que nos encontramos a la hora de resolver un problema de satisfacción de restricciones son, primero, cómo representar el problema, y segundo cómo resolverlo. Por consiguiente en problemas con restricciones no binarias debemos tomar la decisión de convertir el problema a uno binario (o ternario) o bien dejarlo en su representación original. Si convertimos el problema a uno binario (ternario) entonces podremos aplicar todos los algoritmos y heurísticas generados para resolver CSPs binarios (ternarios). Esta opción ha sido mayoritariamente seleccionada por la comunidad científica.

Sin embargo si mantenemos el problema en su representación original, tendremos que usar los algoritmos y heurísticas para restricciones no binarias, encontrándonos aquí con falta de respuestas útiles para el manejo de CSPs no binarios.

Principalmente hay dos técnicas para trasladar las restricciones no binarias a binarias en CSPs discretos:

- La codificación dual, que fue introducida en la comunidad de CSPs por Dechter y Pearl [6], donde la idea básica se capturó de la investigación en bases de datos relacionales [16].
- La codificación de variables ocultas, que procede de Peirce quien, en 1933, probó formalmente en el campo de la filosofía lógica que las relaciones binarias y no binarias tienen el mismo poder de expresividad [18]. Su método para representar una relación no binaria mediante una colección de restricciones binarias formó las bases del método de las variables ocultas.

Además, Rossi y otros probaron que un CSP no binario es equivalente a su transformación dual y oculta bajo varias definiciones de equivalencia, que definiremos a continuación [21].

En el caso de CSPs numéricos o continuos, se pueden transformar las restricciones $n-$arias en restricciones ternarias [14] de forma que se mejora la eficiencia de los algoritmos de consistencia tales como 2-consistencia [7, 9], 2B-consistencia, 3B-consistencia [13] o la consistencia $(3,2)-$relacional. Esta transformación sólo es posible si se permite la introducción de variables auxiliares.

Este artículo lo vamos a clasificar en las siguientes secciones: en la sección 2 se presentan los diferentes conceptos necesarios de CSPs no binarios. En la sección 3 se presentan las diferentes técnicas de transformación de CSPs no binarios a binarios: la codificación dual, la codificación de variables ocultas, la codificación doble y por último un algoritmo de transformación de restricciones no binarias a ternarias para CSPs continuos. En la sección 4 presentamos un método llamado HSA para manejar CSPs no binarios sobre dominios continuos. En la sección 5 presentamos la aplicación de HSA al análisis ROC. Finalmente, en la sección 6 presentamos las conclusiones.

2. Definiciones

En esta sección nos vamos a centrar en los conceptos necesarios y que no están contemplados en el artículo de introducción, para comprender la notación que vamos a utilizar en este trabajo.

La equivalencia entre dos CSPs es fundamental a la hora de transformar un CSP no binario a uno binario para no desvirtuar el problema, pero puede resultar muy ambiguo dependiendo de aspectos tanto sintácticos como semánticos. En [21] se presentan diferentes definiciones de equivalencia.

Una primera definición ingenua de equivalencia

de CSP podría ser:

- *Dos CSPs son equivalentes si ellos tienen las mismas variables y las mismas restricciones.*

Esta propuesta de equivalencia es muy restrictiva. Por ejemplo, esta propuesta no considera la posibilidad de información redundante en las restricciones. Esta redundancia podría provocar, dado un CSP, un conjunto infinito de CSPs distintos, simplemente seleccionando uno dado, y añadiendo información redundante. De esta manera, todos los CSPs resultantes representarían el mismo problema, y por tanto, serían equivalentes. Por lo tanto, la definición dada anteriormente no sería válida.

Para solucionar este problema modificaron la definición de equivalencia para manejar la redundancia de las restricciones adoptando la siguiente definición:

- *Dos CSPs son equivalentes si tienen el mismo conjunto de soluciones.*

Aquí el significado previsto de solución de un CSP es el conjunto de todas las asignaciones de las variables a los dominios de tal forma que se satisfagan todas las restricciones del problema. Esta es la definición usual de equivalencia de los CSPs y la más aceptada por la comunidad científica. Así, todos los CSPs redundantes, con respecto a uno dado, se consideran equivalentes. Sin embargo la redundancia puede ocurrir no sólo en las restricciones sino también en el número de variables, ya que es posible que dos CSPs tengas diferentes variables y que todas ellas se comporten de la misma manera, es decir, sólo difieren los nombres de las variables. Tales problemas, que sintácticamente son diferentes, deberían considerarse equivalentes siempre que la información contenida sea esencialmente idéntica. En otras palabras, lo único que importa es la información contenida en el CSP. De acuerdo a este hecho, Rossi y otros propusieron la siguiente definición de equivalencia de CSPs:

- *Dos CSPs son equivalentes si son mutuamente reducibles.*

Así, dados dos CSPs P_1 y P_2, se dice que P_1 es reducible a P_2 si es posible obtener la solución de P_1 de la solución de P_2, mediante el mapeo de las variables y los valores de un problema a las variables y los valores del otro problema. Esta definición contempla la redundancia de restricciones y de variables y se puede demostrar que es más general que las definiciones previas de equivalencia.

Así, esta definición de equivalencia, se utilizó en [21] para probar la equivalencia de CSPs no binarios y binarios, dando importancia a esta transformación no sólo desde el punto de vista teórico, sino también práctico, ya que muchos algoritmos eficientes sólo se han desarrollado para CSPs binarios. Sin embargo hay que tener en cuenta si es conveniente manejar el problema no binario como tal, o es más conveniente transformarlo a un CSP binario, resolverlo, y finalmente ser capaz de devolver la solución del problema original.

En la literatura de CSPs, la equivalencia entre CSPs no binarios y binarios se ha estudiado de una manera superficial. El primer intento formal para comparar estas dos representaciones deferentes de un mismo problema apareció en 1933, donde Peirce en [18] mostró un ejemplo de una restricción no binaria que no se puede representar mediante restricciones binarias. De esta manera, las restricciones binarias y no binarias parecen no ser equivalentes en general. Más precisamente, las restricciones no binarias parecen ser más potentes que las binarias. Esto ocurre cuando se ponen restricciones sobre el número de variables a utilizar en el problema transformado, o en la dimensión de las variables en la nueva representación.

En el artículo de introducción ya presentamos una definición general de CSP. El problema de satisfacción de restricciones no binarias que vamos a manejar lo podemos definir como un CSP que consta de una terna (X, D, C) donde:

- X es un conjunto de n variables reales $\{x_1, ..., x_n\}$.
- D es un vector de dominios continuos o discretos $D =< D_1, ..., D_n >$. Cada variable x_i tiene un dominio $D_i : [l_i, u_i]$. El dominio, continuo o discreto, de una variable es el conjunto de todos los posibles valores que se le pueden asignar a esa variable.
- C es un conjunto finito de restricciones que la solución debe satisfacer. Cada restricción c_i está definida sobre un conjunto de variables $\{x_1, ..., x_n\}$ que tomarán valores de sus respectivos dominios $D_1, D_2, ...D_n$.

2.1. Ejemplo de CSP Generales

Como ejemplos de CSPs generales, es decir CSPs con restricciones binarias y no binarias con variables acotadas en dominios discretos y continuos, vamos a presentar dos ejemplos presentados en [26] relativo a aspectos de la ingeniería civil en el diseño de construcciones.

El primer ejemplo trata del diseño de un puente, donde el trabajo a realizar puede verse como una combinación de dos tareas: El *diseño conceptual* que define la estructura que va a tener el puente así como sus parámetros, y el *diseño detallado* que encuentra un conjunto de valores para los parámetros en la solución final (ver Figura 1. En el diseño conceptual se manipulan descripciones parciales de la obra a realizar, consistiendo en partes (pilares, cubiertas, arcos del puente, etc) y propiedades (reglas de construcción, requerimientos artísticos y económicos, leyes físicas, etc...).

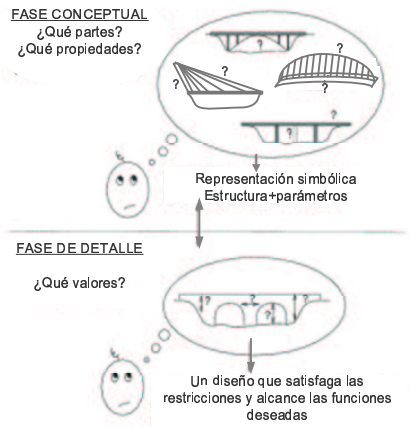

Figura 1. Paso del diseño al CSP.

Una tarea de diseño se presta naturalmente a una formulación de satisfacción de restricciones. Las variables representan las partes y elementos del diseño, mientras que las restricciones expresan las propiedades que las partes deben satisfacer. Resolver tal problema de satisfacción de restricciones significa trasladar la representación simbólica utilizada en el diseño conceptual a los objetos reales: las soluciones al CSP determinan si, y dentro de qué límites, la descripción generada en el diseño conceptual corresponde a una estructura física factible.

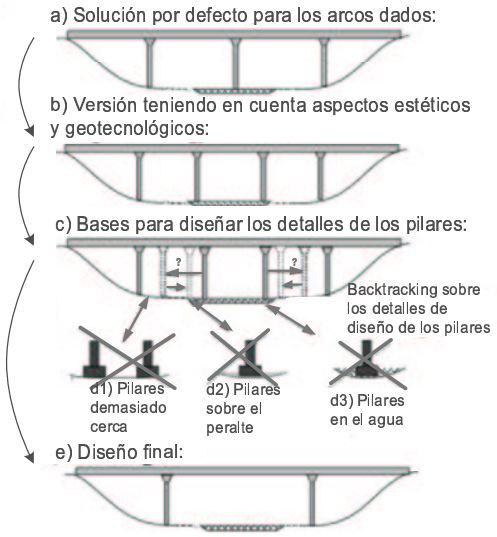

Figura 2. Diferentes fases en el diseño de un puente: (i) cada estado corresponde a un CSP, (ii) la solución a un CSP dado impacta con la naturaleza de otros CSPs. El CSP del estado c) no admite solución si no hay una ubicación definida para los pilares del puente.

Para desarrollar un modelo computacional del proceso de diseño, de acuerdo al paradigma de la satisfacción de restricciones, hay que tener en cuenta diversos factores importantes, tales como:

- El conjunto de variables y restricciones no es independiente de soluciones particulares. Por ejemplo en el problema de diseño de la Figura 2, ir de la solución a) a la solución b) lleva consigo cambiar el número de arcos del puente y por lo tanto el número de variables y de restricciones.
- Las variables pueden tomar valores en dominios discretos (número de pilares, número de arcos, etc...) y dominios continuos (anchura de los arcos, longitud de las cubiertas, altura de los pilares, etc...).
- Se pueden introducir restricciones de inigualdad, restricciones no lineales y por supuesto restricciones que involucren a cualquier número de variables (restricciones no binarias).

→ La primera consideración muestra que el CSP relativo al diseño del problema no está determinado a priori antes de comenzar el proceso de resolución, por lo que a veces es necesario trabajar con problemas dinámicos o incrementales,

donde hay restricciones y variables que se pueden modificar, eliminar o añadir.

→ La segunda consideración nos muestra que hay muchos problemas reales donde se entrelazan dominios tanto discretos como continuos, con la consiguiente complejidad que ello conlleva.

→ La tercera y última consideración se centra en la posibilidad de manejar restricciones complejas como pueden ser restricciones no lineales e inigualdades de cualquier aridad.

Por lo tanto debemos tener en cuenta la existencia de problemas reales donde coexisten variables continuas con variables discretas enmarcadas en restricciones no binarias, lo que genera la necesidad de desarrollar técnicas de satisfacción de restricciones capaces de manejar estos tipos de problemas. Además, desde que los valores numéricos de los parámetros sirven como base de decisiones, es necesario identificar el espectro de alternativas tan exhaustivamente como sea posible, para que el espacio de posibilidades pueda explorarse sistemáticamente.

El siguiente ejemplo que se muestra en la Figura 3, relativo al diseño de edificios, ilustra este punto de una manera más precisa. La estructura del suelo de un edificio está formada por planchas de hormigón sobre vigas de acero. Dependiendo de las medidas de las vigas (longitud (w) y altura (h)), se pueden generar diferentes alternativas de diseño (ver Figura 4), cada una involucrando nuevas variables y restricciones.

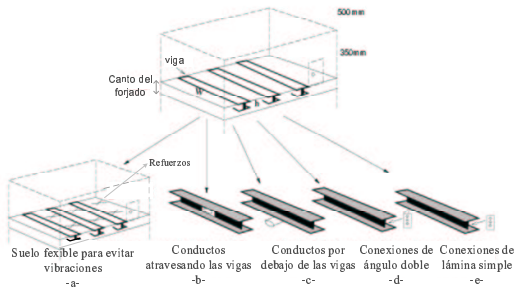

Figura 3. Un problema de diseño presentando varias opciones.

Cuando los sistemas de suelo se hacen cada vez más delgados, las vibraciones aumentan. Esto requiere instalar refuerzos laterales para flexibilizar el suelo (Figura 3, opción -a-). Este es el caso cuando las dimensiones de las vigas caen en las regiones 3 y 4 de la Figura 4.

El *canto del forjado* es el espacio situado entre el techo de una planta y el suelo de la planta superior (ver Figura 3). El canto de forjado tiene con frecuencia una altura fija, y existe una preferencia general de situar los conductos de la ventilación por debajo de las vigas dentro del canto de forjado (falso techo) (ver Figura 3, opción -c-). Sin embargo para vigas relativamente altas, donde la altura (h) es mayor de 500mm (región 4), no hay espacio suficiente para que los conductos pasen por debajo de las vigas, por lo que hay que perforar las vigas (Figura 3, opción -b-) para pasar estos conductos de ventilación.

Por último hay muchas maneras para conectar una viga con otra. La conexión más económica es por medio de láminas simples, que son láminas que se sueldan perpendicular a la viga para conectarse con otra viga o columna (ver Figura 3, opción -e-). Este tipo de conexiones sólo es apropiado para una altura de viga menor de 350mm (regiones 0 y 1). Para alturas de viga más grandes, se aconseja conexiones de ángulo doble (en forma de L) (ver Figura 3, opción -d-) para garantizar más la seguridad (regiones 2, 3 y 4).

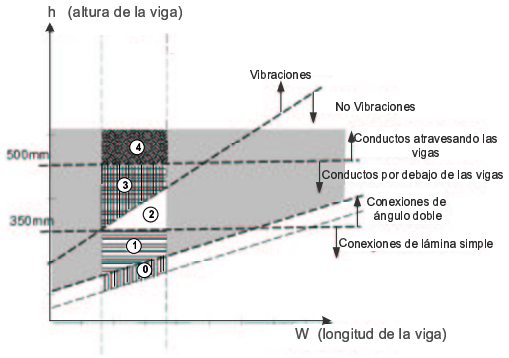

Figura 4. Un problema de diseño presentando varias opciones.

Cuando los valores numéricos de los parámetros sirven como base para tomar decisiones e influyen en el proceso de resolución completo, la identificación de una sola solución que satisfaga el conjunto de restricciones es con frecuencia insuficiente. En el ejemplo de la Figura 3, un método matemático tradicional (análisis numérico, métodos aleatorios) generalmente identificaría una solución dentro de las regiones 0,1,2,3 y 4. Esto obliga al diseñador a perder mejores alternativas de diseño.

Para tales aplicaciones, es de crucial importancia proporcionar al usuario, o a otras partes del programa, los rangos o intervalos de valores posibles tan completos y correctos como sea posible, lo cual permite tomar decisiones racionales. En este ejemplo resulta importante la consistencia global (ver artículo de introducción), ya que la altura y el espacio de las vigas están enlazados con otras variables (espesor de la plancha del suelo) mediante diferentes ecuaciones, incluso no lineales. La región 0 resulta ser eventualmente inconsistente con estas restricciones, pero esto no siempre es detectable utilizando solamente técnicas de consistencia local, permitiéndole así al diseñador seleccionar un punto dentro de la region 0 y tomar nuevas decisiones sobre las bases de una estructura no realizable físicamente.

3. Técnicas de Transformación de CSPs no binarios

3.1. Codificación Dual

En la codificación dual, las restricciones del problema original se transforman en variables del nuevo problema y viceversa. Las variables que se generan de las restricciones originales las llamaremos *variables duales*, y a las variables del problema original le llamaremos *variables originales*. El dominio de cada variable dual es el conjunto de tuplas que satisfacen la restricción original que la genera. Además, hay una restricción binaria entre dos variables duales v_c y $v_{c'}$ si y solamente si las restricciones originales c y c' comparten una o más variables. Llamaremos a las nuevas restricciones binarias como *restricciones duales*. Las nuevas restricciones duales prohíben pares de tuplas donde las variables compartidas tomen valores diferentes.

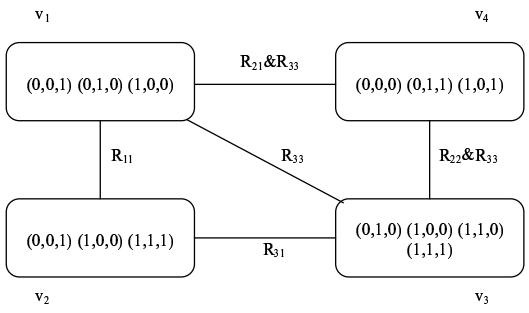

Figura 5. Ejemplo de codificación dual de un CSP no binario.

Ejemplo. Consideremos el siguiente ejemplo, tomado de [29], con seis variables con dominios 0-1 y cuatro restricciones: $x_1 + x_2 + x_6 = 1$, $x_1 - x_3 + x_4 = 1$, $x_4 + x_5 - x_6 \geq 1$ y $x_2 + x_5 - x_6 = 0$. La codificación dual representa este problema con cuatro variables duales, una para cada restricción. Los dominios de estas variables duales son las tuplas que satisfacen las respectivas restricciones. Por ejemplo, la variable dual v_3 asociada a la tercera restricción, $x_4 + x_5 - x_6 \geq 1$, tiene como dominio $(0,1,0), (1,0,0), (1,1,0), (1,1,1)$, ya que estas son las tuplas de valores para (x_4, x_5, x_6) que satisfacen la restricción. La codificación dual del problema se muestra en la Figura 5. R_{ij} es la restricción binaria sobre un par de tuplas que se satisface si y sólo si el $i-$ésimo elemento de la primera tupla es igual al $j-$ésimo elemento de la segunda tupla. Entre v_3 y v_4 hay una restricción de compatibilidad para asegurar que las dos variables originales en común, x_5 y x_6 tienen los mismos valores. Esta restricción permite solamente aquellos pares de tuplas que concuerdan con los elementos segundos y terceros, es decir $(1,0,0)$ para v_3 y $(0,0,0)$ para v_4.

3.2. Codificación de Variables Ocultas

En la codificación de variables ocultas, el conjunto de variables está formado por todas las variables del CSP no binario original más un nuevo conjunto de variables duales que llamaremos *variables ocultas*. Al igual que la codificación dual, cada variable oculta v_c corresponde a una restricción en el problema original. De nuevo, el dominio de cada variable oculta consta de las tuplas que satisfacen la restricción original. Para cada variable oculta v_c, hay una restricción binaria entre la variable v_c y cada una de las variables originales x_i que están involucradas en la correspondiente restricción original c. Cada restricción especifica que la tupla asignada a v_c es consistente con el valor asignado a x_i.

Considerando de nuevo el ejemplo anterior, donde el problema consta de seis variables y cuatro restricciones no binarias. En la codificación de variables ocultas hay, además de las seis variables originales 0-1, cuatro variables duales con los mismos dominios que en la codificación dual. Por ejemplo, la variable dual asociada con la tercera restricción $x_4 + x_5 - x_6 \geq 1$, tiene como dominio $(0,1,0), (1,0,0), (1,1,0), (1,1,1)$.

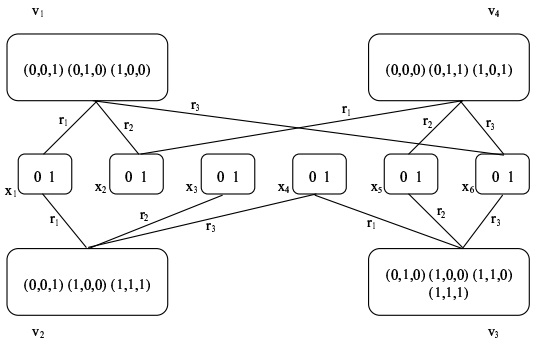

Figura 6. Ejemplo de codificación de variables ocultas de un CSP no binario.

Además ahora hay restricciones de compatibilidad entre v_3 y x_4, entre v_3 y x_5 y entre v_3 y x_6, ya que estas son las variables que están involucradas en la tercera restricción. La codificación de variables ocultas de este problema se muestra en la Figura 6. La restricción binaria r_i actúa sobre una tupla y un valor, dando verdadero si y sólo si el elemento $i-$ésimo de la tupla es igual al del valor. Por ejemplo, la restricción de compatibilidad r_1 entre v_3 y x_4 es cierta si y sólo si el primer elemento de la tupla asignada a v_3 es igual al valor de x_4.

3.3. Codificación Doble

En [29] se propone una codificación llamada *codificación doble* en la cual se mezclan tanto la codificación dual como la codificación de variables ocultas. Al igual que la codificación de variables ocultas, la codificación doble tiene tanto variables originales como variables duales (ocultas). Además se mantienen ambos tipos de restricciones: las restricciones entre variables duales, (como en la codificación dual), y las restricciones entre variables duales y variables originales, (como en la codificación de variables ocultas). En la codificación doble, se tiene una poda extra de la alcanzada en la codificación dual. Además se puede ramificar sobre las variables originales como en la codificación de las variables ocultas, mediante heurísticas de ramificación que pueden ser capaces de comportarse mejor sobre estas variables que sobre las duales que pueden tener potencialmente grandes dominios.

Dado el grafo de restricciones correspondiente a la codificación de variables ocultas, se seleccionan los caminos de longitud 2 entre cada par de variables duales v_c y $v_c^{'}$. Estos caminos se eliminan (si no forman parte de otro camino) y se reemplazan por una restricción formada por la composición de las relaciones de los arcos borrados. Cuando alguna de las variables originales se queda aislada del resto en el grafo de restricciones o conectada con sólo una variable dual, se puede eliminar con seguridad. Aplicando estas transformaciones de simplificación a cada par de variables duales, transformaríamos la codificación de variables ocultas en la codificación dual. Sin embargo a veces resulta conveniente quedarse en un punto intermedio de manera que se pueda aprovechar las ventajas de cada codificación.

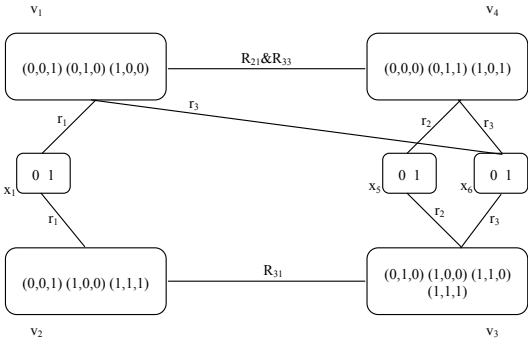

Figura 7. Paso intermedio entre codificación de variables ocultas y codificación dual de un CSP no binario.

Por ejemplo, consideremos la codificación de variables ocultas de la Figura 6. Tomemos el par de variables duales v_2 y v_3. La variable x_4 es la única variable que enlaza, como camino de longitud 2, las variables v_2 y v_3. Se eliminan las restricciones r_3 y r_1 que conectan con x_4 y se añade una nueva relación R_{31} que es la composición de las dos relaciones r_3 y r_1. Como x_4 se queda aislada del resto del grafo, se elimina sin mas. A continuación tomamos otro par de variables duales, por ejemplo v_1 y v_4. Hay dos caminos de longitud dos entre v_1 y v_4. El camino entre v_1 y v_4 vía x_2 tiene las restricciones r_2 y r_1, por lo que induce una restricción R_{21} entre v_1 y v_4 eliminándose el camino anterior. De la misma manera el camino entre v_1 y v_4 vía x_6 tenía ambas restricciones etiquetadas como r_3, por lo que induce una restricción R_{33} entre v_1 y v_4. En este caso no se elimina el camino anterior porque sus arcos forman parte de otros caminos. Por lo tanto la restricción inducida entre las variables v_1 y v_4 será la unión de ambas $R_{21}\&R_{33}$. Así ahora podemos borrar la variable x_2 y también la variable x_3 ya que sólo está conectada con una variable dual v_2. Hasta este momento el grafo de restricciones resultante es el que se muestra en la Figura 7.

3.4. Inconvenientes

La transformación de un problema no binario en uno binario no está en absoluto exento de inconvenientes, ya que en muchos casos las restricciones $n-$arias pierden una parte crucial de su claridad expresiva cuando se traslada a conjuntos de restricciones binarias. Además, la eficiencia de transformar restricciones no binarias en binarias no se ha estudiado adecuadamente. En la práctica, esta transformación puede ser poco eficiente debido a su alto coste computacional haciendo que con frecuencia esta transformación sea impracticable [12, 4]. El proceso de transformación genera nuevas variables, las cuales pueden tener dominios muy grandes, causando unos requerimientos de memoria extra para los algoritmos que lo resuelven. Por ello, en algunos casos, resolver el problema binarizado puede resultar muy ineficiente [2]. Además, esta binarización forzada puede generar una formulación poco natural causando unas dificultades extras para las interfaces de los resolvedores de restricciones con el usuario humano [5].

Por ejemplo un serio inconveniente de estas transformaciones es que si las restricciones no binarias están representadas implícitamente, las codificaciones anteriores podrían tener una complejidad exponencial tanto en espacio como en tiempo. Por ejemplo, si una restricción no binaria sobre $x_1, ..., x_n$ se representa mediante una función $f(x_1, ..., x_n)$, la cual devuelve verdadero si la asignación de las variables satisfacen la restricción, en la codificación dual (oculta), cada tupla consistente en una restricción se transforma en un valor del dominio de la correspondiente variable dual (oculta). Esto puede llevar consigo un número exponencial de pasos para encontrar esas tuplas así como una cantidad exponencial de espacio para almacenar estas tuplas en el dominio de la variable dual (oculta).

3.5. Transformación a restricciones ternarias

Es conveniente en CSPs numéricos transformar las restricciones $n-$arias en restricciones ternarias de forma que se mejore la eficiencia de los algoritmos de consistencia como 2-consistencia [7, 9], 2B-consistencia, 3B-consistencia [13] y consistencia $(k, k-1)$-relacional [26] donde k es la aridad de las restricciones.

En general, cualquier expresión matemática construida mediante operadores unarios y binarios puede reescribirse de forma ternaria introduciendo una variable auxiliar para cada resultado intermediario generado mediante un operador binario. Este método genera muchas variables auxiliares. Por lo tanto no se han generado algoritmos prácticos para llevar a cabo la tarea de transformación de un CSP numérico en uno ternario de forma automática, por lo que esta transformación se lleva a cabo con frecuencia a mano [14].

Los algoritmos de consistencia mencionados anteriormente dependen fundamentalmente del número de variables del CSP, por lo que es necesario el uso de algoritmos que durante la transformación generen pocas variables auxiliares. Por la misma razón, es interesante comprobar la posibilidad de eliminar variables innecesarias del CSP original, ya que muchas veces, cuando se formaliza un problema se utilizan variables intermediarias y constantes para hacer el CSP más legible y reutilizable. Sin embargo, cuando se trata de obtener la consistencia del problema, puede ser beneficioso la eliminación de tales variables.

En [26] se demuestra que para un CSP de aridad k, dándose ciertas restricciones de convexidad parcial, la consistencia $(k, k-1)-$relacional es equivalente a la consistencia global, es decir todas las soluciones se pueden encontrar sin la necesidad de backtracking. Forzar la consistencia $(k, k-1)-$relacional requiere un algoritmo de complejidad $O(n^{2k-1})$ donde n es el número de variables del CSP. Debido a que esta complejidad es exponencial en k, transformar un CSP numérico en uno de baja aridad antes de llevar a cabo la consistencia $(k, k-1)-$relacional tiene la posibilidad de acelerar el cálculo.

En [14], Lottaz presenta un método para transformar los CSPs numéricos en términos de restricciones ternarias introduciendo variables auxiliares. Veamos a continuación la forma en la que propone llevar a cabo esta transformación.

Los CSPs numéricos cuyas restricciones son expresiones matemáticas que utilizan operadores unarios y binarios pueden transformarse en restricciones de baja aridad introduciendo variables auxiliares para los resultados intermediarios de todos los operadores binarios de una restricción. En la Figura 8 podemos observar como una restricción 5-aria se transforma en un conjunto de restricciones ternarias.

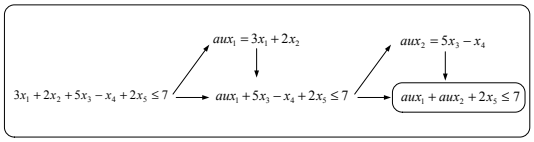

Figura 8. Transformación de una restricción 5-aria en ternarias.

Así la aridad de las restricciones se puede reducir a tres a cambio de incrementar el número de variables (variables auxiliares) y de restricciones (las definiciones de variables auxiliares). Una restricción que define una variable auxiliar que actualmente ayuda a reducir la aridad de otra restricción tiene al menos aridad tres. En caso contrario, la variable auxiliar reemplazaría una expresión que depende solamente de una variable por una nueva variable, por lo que no reduciría la aridad de ninguna restricción. De esta manera, un CSP numérico transformado de esta manera que hemos resumido puede reducirse reemplazando operadores binarios por variables auxiliares.

Por lo tanto, por un lado, decrementando la aridad del CSP antes de llevar a cabo la consistencia $(k, k-1)-$relacional reduce la complejidad computacional y por el otro lado hay un intercambio entre decrementar la aridad e incrementar el número de variables del CSP para alcanzar una aridad menor. De esta manera para decidir si merece la pena transformar el CSP en uno de aridad menor, tenemos que estimar el número de variables que se crearán en la transformación. Teniendo en cuenta que la complejidad computacional de la consistencia $(k, k-1)-$relacional es exponencial con respecto a k, el autor solamente considera el caso cuando la aridad se reduce al máximo, es decir, a tres.

En el peor caso, transformar un CSP numérico de aridad $k > 3$ en forma ternaria requiere la adición de $O(m)$ variables auxiliares, donde m es el número de operadores binarios en el CSP. En este caso la complejidad $O((n+m)^5)$ para la consistencia $(3, 2)-$relacional sobre el conjunto de restricciones transformadas se compara con la complejidad $O(n^{2k-1})$ para la consistencia $(k, k-1)-$relacional del CSP original. En problemas prácticos la influencia exponencial de k puede ser que tenga un peso mucho mayor que la influencia polinomial de m.

Dado que cualquier CSP numérico especificado mediante expresiones matemáticas con operadores unarios y binarios puede ser transformado en un CSP ternario, se han generado varios algoritmos de consistencia para CSP numéricos con restricciones ternarias [7, 9, 13].

4. CSPs no binarios con dominios continuos

Como ya hemos apuntado, la mayoría de la investigación realizada se ha centrado en problemas binarios debido principalmente a la sencillez que supone comparado con las no binarias y por la posibilidad de transformación a uno binario equivalente [21]. Además cuando se manejan las restricciones en su formulación original (no binarias) y éstas son restricciones globales, la complejidad se incrementa sustancialmente, e incluso se pueden hacer impracticable los algoritmos de preproceso que vimos en el capítulo de introducción, para alcanzar ciertos niveles de consistencia. Además, en muchas aplicaciones reales se intenta extraer del CSP la máxima información posible, por lo que no es suficiente obtener la consistencia del problema, sino que a veces es interesante obtener cierta información sobre las soluciones del CSP, los dominios mínimos de ciertas variables del problema, etc.

En esta sección presentamos un algoritmo para la resolución de CSPs no binarios sobre dominios continuos que obtiene además la consistencia global del problema. Este algoritmo lo llamaremos HSA [23, 25] (Hyper-polyhedron Search Algorithm) o Algoritmo de Búsqueda Hiperpoliédrica, debido a que el algoritmo lleva a cabo su ejecución mediante un hiper-poliedro o polítopo. Este algoritmo lo podemos considerar como un resolutor de CSPs numéricos, ya que los dominios de las variables son intervalos en $\mathbb{R}$ y las restricciones se representan como expresiones matemáticas. El objetivo de este algoritmo no se centra solamente en la obtención de la consistencia global del problema sino que trata de alcanzar los objetivos antes comentados para obtener la máxima información posible del problema.

A grandes rasgos este algoritmo trabaja de la siguiente manera: dados los dominios de las variables, el algoritmo genera un polítopo a partir del producto cartesiano de dichos dominios. Este polítopo convexo mantendrá en todo momento el conjunto de soluciones del problema, y se irá actualizando a medida que las restricciones no binarias van siendo analizadas.

4.1. Especificación general de HSA

HSA lo consideramos inicialmente como un resolvedor o resolutor de CSPs estático, donde hay un conjunto inicial de restricciones que la solución debe de satisfacer. En la Figura 9 se presenta el comportamiento general de HSA. Este esquema se divide en cuatro pasos: generación del polítopo; estudio de la consistencia con las restricciones de inigualdad; actualización del polítopo; y estudio de la consistencia con las restricciones de desigualdad.

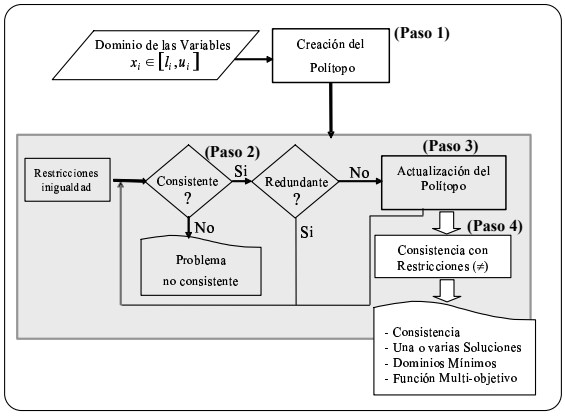

Figura 9. Especificación Gráfica de HSA.

El paso 1 siempre se lleva a cabo en todos los CSPs. El paso 2 y 4 se llevan a cabo cuando existen restricciones de inigualdad y de desigualdad respectivamente. El paso 3 se lleva a cabo cuando la restricción en estudio es consistente y no redundante.

HSA genera un polítopo inicial (*Paso* 1) mediante el producto cartesiano de los límites de los dominios de las variables $(D_1 \times D_2 \times ... \times D_n)$ cuyos vértices son: $v_1 = (l_1, l_2, ..., l_n), ..., v_i = (l_1, l_2, ..., l_j, u_{j+1}, ..., u_n), ..., v_{2^n} = (u_1, u_2, ..., u_n)$. Estos vértices se almacenan en una lista llamada $ListV$. Una vez generado el polítopo, HSA pasa a estudiar la consistencia de las restricciones del problema. Para ello, HSA analiza primero las restricciones de inigualdad $(\leq)$ y a continuación si el problema es consistente, estudia las restricciones de desigualdad $(\neq)$. Para cada restricción de inigualdad, HSA lleva a cabo la comprobación de la consistencia (*Paso* 2). Para ello, el algoritmo comprueba si los vértices del poliedro satisfacen la restricción en estudio, almacenando los vértices factibles en una lista auxiliar llamada L_{si} y los no factibles en una lista llamada L_{no}. Si al menos un vértice es consistente, es decir $L_{yes} \neq \phi$, entonces la restricción es consistente. Si la restricción de inigualdad no es consistente, es decir $L_{yes} = \phi$, entonces HSA devuelve *'problema no consistente'* y finaliza la ejecución. Sin embargo si la restricción es consistente, HSA estudia si la restricción es o no redundante, actualizando el polítopo (*Paso* 3) en el caso de que no lo fuera $(L_{no} \neq \phi)$. La actualización sólo se lleva a cabo cuando la restricción es consistente y no redundante, ya que es cuando únicamente la inigualdad intersecta al polítopo actual. Cuando todas las restricciones de inigualdad se han analizado y el problema es consistente, se lleva a cabo la comprobación de la consistencia con las restricciones de desigualdad (*Paso* 4). En este caso el polítopo no se actualiza, ya que las restricciones de desigualdad son hiperplanos que pueden intersectar o no al poliedro, pero a lo sumo sólo eliminan del polítopo un hiperplano, por lo que hay que estudiar las consecuencias de dicha intersección [24].

Una vez finalizada la comprobación de la consistencia de las restricciones de inigualdad y de desigualdad, HSA devuelve al usuario información relevante para el usuario como la consistencia del problema, una, varias o las soluciones extremas del problema, los dominios mínimos de las variables, etc.

Además, cuando HSA finaliza su comportamiento estático, como clásico resolvedor de CSPs, donde sólo se han estudiado las restricciones de entrada del problema, HSA puede estudiar la consistencia de nuevas restricciones que se inserten en el problema de forma incremental. Este estudio de la consistencia de las nuevas restricciones se lleva a cabo sin la necesidad de iniciar todo el proceso de nuevo, ya que tan sólo se estudia la consistencia de las nuevas restricciones insertadas con el polítopo resultante en la etapa de CSP clásico.

Teorema El algoritmo HSA es un algoritmo completo y correcto [22].

La complejidad espacial de HSA viene dada por la siguiente expresión:

$$O(max\{2^n, nc_{\leq}, c_{\leq}^2\}) \equiv O(2^n)$$

La complejidad temporal es:

$$O(2^n) + c_{\leq} \cdot O(2^n) + c_{\neq} \cdot O(2^n) \equiv O(2^n).$$

5. Aplicación al análisis ROC

En esta sección vamos a presentar el planteamiento general de un problema de decisión que final-

mente se modela como un CSP no binario sobre dominios continuos. Esto ocurre en muchos problemas reales donde hay que tomar decisiones para llevar a cabo una acción u otra. Para ello se utilizan los llamados *clasificadores*, que hacen predicciones sobre dichas decisiones utilizando para ello experiencias previas. Sin embargo, debido a que las predicciones pueden ser erróneas, es importante conocer cual es su efecto, cuando éstas son incorrectas. En muchas situaciones los errores tienen distintas consecuencias. Algunos errores tienen unos costes más elevados que otros, especialmente en el campo de la diagnosis. Por ejemplo, un tratamiento o diagnosis erróneo puede tener diferentes costes y peligros dependiendo del tipo de error que se ha producido. Obviamente, los costes de cada clasificación errónea son dependientes del problema, pero casi nunca se da el caso de que ellos sean uniformes para un solo problema. Consecuentemente, la precisión o el error no es generalmente la mejor manera para evaluar la calidad de un clasificador o de un algoritmo de aprendizaje.

El aprendizaje sensible al coste es una generalización más realista del aprendizaje predictivo, y los modelos sensibles al coste permiten una mejor toma de decisiones. La calidad de un modelo se mide en términos de minimización de coste más que en términos de minimización de errores. Cuando se proporcionan a priori las matrices de coste, es decir, antes de que el aprendizaje tenga lugar, las matrices tienen que explotarse completamente para obtener modelos que minimicen el coste. Sin embargo, en muchas circunstancias, los costes no son conocidos a priori o los modelos ya están seleccionados. El análisis ROC (Receiver Operating Characteristic) [19, 30, 20] ha demostrado ser muy útil para la evaluación de los clasificadores cuando la matriz de coste no era conocida en la construcción de los clasificadores. El análisis ROC proporciona herramientas para seleccionar un conjunto de clasificadores que se comportarían óptimamente y para rechazar algunos otros clasificadores menos útiles. Para hacer esto, se construye la envoltura convexa de todos los clasificadores, dando una curva que junto a los ejes genera un polítopo convexo. esto se puede realizar de manera sencilla para clasificadores binarios (2 clases) pero hasta el momento no se ha presentado una solución para más de dos clases.

Como veremos, el análisis ROC puede ser modelado como un CSP numérico y no binario que puede ser resuelto por el algoritmo completo HSA mientras que es difícil resolverlo mediante los resolutores tradicionales debido a que estos trabajan principalmente con problemas discretos y con restricciones binarias donde su objetivo se centra en el estudio de la consistencia y en la obtención de una o varias soluciones del problema. Sin embargo en el problema que estamos considerando, se necesitan obtener el espacio de soluciones factible y esto lo consigue HSA mediante la obtención de todas las soluciones extremas que se necesitan para obtener el hiper-poliedro resultante y obtener así el volumen que ocupa la envoltura convexa del espacio ROC de clasificadores de n clases pudiendo ser $n > 2$ [8].

5.1. Utilización de HSA para la obtención de polítopos ROC

En esta sección, presentamos la aplicabilidad que tiene el algoritmo completo HSA para llevar a cabo la extensión real del área bajo la curva ROC (Area Under the Curve, AUC) al volumen bajo la superficie ROC que llamaremos VUS (Volume Under ROC Surface), mostrando cómo generar el hiper-poliedro que englobe a todos los clasificadores. Además compararemos el VUS real obtenido mediante HSA con aproximaciones o extensiones de la AUC para más de dos clases.

El análisis ROC y la medida AUC se han usado exhaustivamente en el ámbito de la toma de decisiones en medicina [11, 17], así como en el área de la extracción de conocimiento, en las herramientas de minería de datos, el reconocimiento de patrones [1] o la ciencia en general [30].

En el caso trivial, un clasificador de dos clases forma una curva ROC compuesta por cuatro segmentos que se generan con los siguientes puntos (ver Figura 10): el punto dado por el clasificador (0.46,0.32) que proviene de los errores de clasificar como a cuando realmente es b ($a \rightarrow b$) y de clasificar como b cuando realmente es a ($b \rightarrow a$), los dos puntos que representan los clasificadores triviales (es decir, el clasificador que siempre predice la clase 0 y el clasificador que siempre predice la clase 1) y el punto origen. En la Figura 10 presentamos de forma general la representación de un clasificador con dos clases, donde posteriormente explicaremos el significado de los polígonos generados. Estos polígonos generan un área que puede ser calculada y que lógicamente variará entre 0.5 (área mínima) y 1 (área máxima). El área del polígono superior, que hemos llamado AUC, se ha convertido en una mejor alternativa que la precisión o el error para evaluar clasificadores, ya

que obtiene la bondad de un clasificador para diferentes distribuciones de las clases.

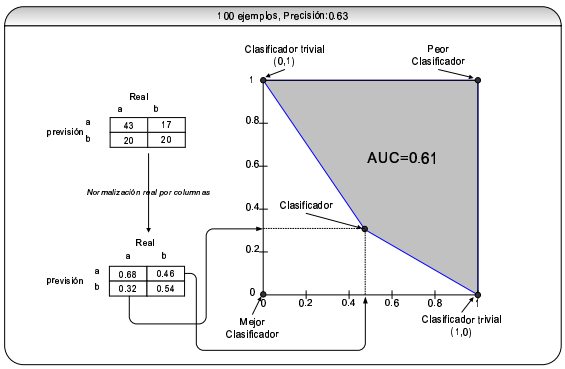

Figura 10. Representación de un clasificador con dos clases.

Sin embargo, la aplicabilidad del análisis ROC y del AUC solamente se ha mostrado viable para problemas con dos clases. Aunque teóricamente el análisis ROC puede extenderse para manejar problemas multi-dimensionales [28], sin embargo en la práctica no se ha podido utilizar debido a la dificultad de la formulación de los clasificadores triviales. El principal inconveniente es la alta dimensionalidad.

No obstante, a pesar de la dificultad, vamos a ver que es posible llevar a cabo el análisis ROC para más de dos clases y el cálculo de la AUC o, más precisamente, el volumen bajo la superficie ROC (VUS). Sin embargo, en la literatura, tanto los clasificadores triviales para más de dos clases, como el volumen mínimo y máximo no se han identificado hasta la fecha.

En esta sección resumimos la forma de calcular los clasificadores triviales, y el VUS mínimo y máximo para clasificadores de más de dos clases. Nosotros compararemos experimentalmente el VUS real obtenido utilizando el algoritmo HSA, con las extensiones de AUC obtenidas por Hand&Till [10].

Veamos a continuación las equivalencias que se generan cuando pasamos del estudio de clasificadores de dos clases a tres clases:

- El polígono generado bajo la curva ROC de los clasificadores de dos clases, se transforma en un hiper-poliedro de 6 dimensiones para los clasificadores de tres clases, por lo que su visión gráfica se hace imposible.
- El área bajo la curva ROC (AUC) se calcula con los cuatro puntos necesarios para generar el polígono en clasificadores de dos clases. En cambio, para tres clases, se transforma en el volumen bajo la superficie (VUS) de un hiper-poliedro de 6 dimensiones del cual se necesitan todos los puntos extremos para dicho cálculo. Pasemos, en primer lugar, a calcular el volumen máximo y mínimo.

5.2. Volumen máximo bajo la superficie (VUS) para tres clases

El volumen máximo debería representar todos los posibles clasificadores.

Para obtener todos los posibles clasificadores, deberemos resolver el siguiente CSP continuo:

- Variables: $x_1, x_2, x_3, x_4, x_5, x_6$;
- Dominios continuos: $x_i \in [0,1], \forall i : 1..,6$;
- Restricciones:
  - $x_3 + x_5 \leq 1$;
  - $x_1 + x_6 \leq 1$;
  - $x_2 + x_4 \leq 1$;

Un punto dentro del hiper-poliedro resultante representa a un clasificador con tres clases.

Dados 6 valores bajo una distribución uniforme entre 0 y 1, representado por U(0,1), tenemos que la probabilidad de que un punto formado por estos 6 valores satisfaga las tres restricciones anteriores es:

$VUS^{max} = P(U(0,1)+U(0,1) \leq 1) \cdot P(U(0,1)+U(0,1) \leq 1) \cdot P(U(0,1)+U(0,1) \leq 1) = P(U(0,1)+U(0,1) \leq 1)^3$.

Es fácil ver que la probabilidad de que la suma de dos números aleatorios bajo la distribución uniforme U(0,1) sea menor que 1 es exactamente $1/2$, es decir,

$$P(U(0,1)+U(0,1) \leq 1) = 1/2$$

Por lo tanto, el volumen máximo teórico es:

$$VUS^{max} = (1/2)^3 = 1/8$$

Sin embargo, ¿cuáles son los puntos cuya envoltura convexa componen este volumen?

Para ello resolvemos el CSP continuo anterior mediante el algoritmo HSA obteniendo 41 puntos extremos, donde el volumen que genera la envoltura convexa de estos 41 puntos, utilizando Qhull [3], es 1/8, tal y como hemos visto teóricamente (Vease también [8]).

5.3. Volumen mínimo bajo la superficie (VUS) para tres clases

Veamos la manera de obtener el VUS mínimo. Sin pérdida de generalidad podemos construir clasificadores triviales como se muestra en la Figura 11 (izquierda), donde $h_a + h_b + h_c = 1$.

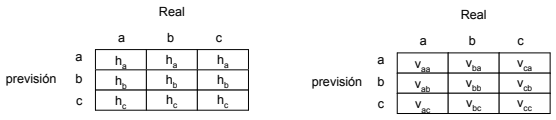

Figura 11. Clasificador trivial y clasificador cualquiera.

Esto, obviamente incluye los tres clasificadores triviales extremos: 'todo es a', 'todo es b' y 'todo es c'.

Dado un clasificador cualquiera como el mostrado en la Figura 11 (derecha), nosotros podemos descartar este clasificador si y sólo si:

$$\exists h_a, h_b, h_c \in \mathbb{R}^+ \ : \ h_a + h_b + h_c = 1 \text{ y además}$$

$$v_{ba} \geq h_a;\ v_{ca} \geq h_a;\ v_{ab} \geq h_b;\ v_{cb} \geq h_b;\ v_{ac} \geq h_c;\ v_{bc} \geq h_c$$

De aquí podemos derivar el siguiente teorema [8]:

Teorema. Un clasificador $(x_1, x_2, x_3, x_4, x_5, x_6)$ se puede descartar si y sólo si $r_1 + r_2 + r_3 \geq 1$, donde $r_1 = min(x_1, x_2)$, $r_2 = min(x_3, x_4)$ y $r_3 = min(x_5, x_6)$

Con las propiedades previas, nosotros solamente tenemos que calcular el espacio de aquellos clasificadores que cumplen la condición de que $r_1 + r_2 + r_3 \geq 1$, donde $r_1 = min(x_1, x_2)$, $r_2 = min(x_3, x_4)$ y $r_3 = min(x_5, x_6)$, para así obtener el volumen mínimo correspondiente a la ausencia total de información.

De esta manera el CSP no binario y continuo que deberemos resolver mediante el algoritmo HSA es el siguiente:

- Variables: $x_1, x_2, x_3, x_4, x_5, x_6, r_1, r_2, r_3$;
- Dominios continuos: $x_i \in [0,1], i$: 1.,6 y $r_j \in [0,1], j = 1,2,3$;
- Restricciones:
  - $x_3 + x_5 \leq 1$
  - $x_1 + x_6 \leq 1$
  - $x_2 + x_4 \leq 1$
  - $r_1 + r_2 + r_3 \geq 1$, donde $r_1 = min(x_1, x_2)$, $r_2 = min(x_3, x_4)$ y $r_3 = min(x_5, x_6)$

Este CSP no binario se resuelve mediante HSA, obteniendo 25 puntos, donde el volumen que genera la envoltura convexa de estos 25 puntos es 1/180, coincidiendo con el que experimentalmente obtenemos mediante el método de Monte Carlo [8].

Resumiendo, tenemos el siguiente VUS para tres clases:

	Teórico	Monte Carlo	HSA
VUS^{max}	1/8	0.12483	0.125
VUS^{min}	1/180	0.555523	0.55555

5.4. Clasificadores Triviales

A pesar de los buenos resultados obtenidos, parece intuitivo (o al menos así ocurre en clasificadores con dos clases) que si cogemos el mínimo y le añadimos el punto origen (el mejor clasificador) deberíamos obtener el máximo. Sin embargo esto no ocurre, ya que con el mínimo obtenido con un volumen de 1/180 y añadiendo el (0,0,0,0,0,0) obtenemos 10 puntos con un volumen de 1/120 que es muy inferior al 1/8 obtenido como máximo. Esto parece contradictorio, ya que se supone que tener el mejor clasificador daría el máximo. Lógicamente si se tiene un clasificador (0,0,0,0,0,0), cualquier clasificador que tenga un valor mayor de 0 para cualquier coordenada es descartable y, lógicamente, esto debería dar 1/8.

La cuestión es que cuando se añade un clasificador deberíamos ver las condiciones que forma. El

clasificador perfecto genera las ecuaciones de descarte $x_i \geq 0, \forall i = 1.,6$, que son ecuaciones nulas, ya que las variables están acotadas en dominios $[0, 1]$.

De esta manera dado un clasificador C_1, nos podemos plantear la siguiente pregunta ¿qué podemos descartar?

Podremos descartar cualquier clasificador C_2 tal que supere a C_1 combinado con los triviales, es decir, que C_2 tenga cada valor mayor en cada una de las casillas.

En realidad lo que tenemos que mirar es la combinación lineal de los tres triviales con el que tenemos:

$h_a \cdot (1,1,0,0,0,0) + h_b \cdot (0,0,1,1,0,0) + h_c \cdot (0,0,0,0,1,1) + h_d \cdot (z_{ba}, z_{ca}, z_{ab}, z_{cb}, z_{ac}, z_{bc})$

que lo podemos descartar si:

$$\exists h_a, h_b, h_c, h_d \in \mathbb{R}^+ \ : \ h_a + h_b + h_c + h_d = 1$$

y además

$$v_{ba} \geq h_a + 0 + 0 + h_d \cdot z_{ba}$$

$$v_{ca} \geq h_a + 0 + 0 + h_d \cdot z_{ca}$$

$$v_{ab} \geq 0 + h_b + 0 + h_d \cdot z_{ab}$$

$$v_{cb} \geq 0 + h_b + 0 + h_d \cdot z_{cb}$$

$$v_{ac} \geq 0 + 0 + h_c + h_d \cdot z_{ac}$$

$$v_{bc} \geq 0 + 0 + h_c + h_d \cdot z_{bc}$$

Esto da lugar a un CSP no binario y continuo con 10 incógnitas que deberemos resolver mediante el algoritmo HSA.

Del resultado obtenido, nos quedamos con las 6 variables de todos los puntos obtenidos, pudiendo calcular el volumen que genera la envoltura convexa de todos ellos.

Veamos a continuación una tabla comparativa, en la que evaluamos, para distintos ejemplos, nuestra herramienta llamada HSA+QHULL (HSA), con otras aproximaciones al VUS como Macro-average (MAC), 1-P Trivial (TRIV), y el método experimental que se generó mediante el método de Monte Carlo (MC).

	MC	**HSA**	**MAC**	**TRIV**
(0,0,0,0,0,0)	0.125	0.125	1	1
(1,1,1,0,0,0)	0.005	0.005	0	0.333
(0.5,0.5,0,0,0,0)	0.032	0.034	0.666	0.666
(0.5,0,0.5,0,0,0)	0.032	0.031	0.666	0.666
(0.7,0.7,0.7,0,0,0)	0.007	0.008	0.25	0.25
(0.33,0,0.33,0,0,0)	0.052	0.052	0.777	0.777

6. Conclusiones

Gran parte de los problemas de satisfacción de restricciones se pueden modelar de forma natural como problemas de satisfacción de restricciones no binarias. Sin embargo, los investigadores centran su atención en los CSPs binarios, por el hecho de que todo CSP no binario se puede transformar en uno binario. Las codificaciones dual y de variables ocultas son los dos métodos más importantes para transformar las restricciones no binarias en binarias en los CSPs discretos. En el caso de CSPs numéricos la transformación de restricciones no binarias a ternarias es la técnica más utilizada. Estas técnicas de transformación son muy útiles en determinados problemas, ya que tienen un coste relativamente bajo, y se pueden utilizar todas las herramientas existentes para el manejo de CSPs binarios. Sin embargo en determinados casos, estas técnicas de transformación no son válidas o se vuelven ineficientes.

Por lo tanto a veces es conveniente resolver el problema en su formulación original, manteniendo así toda la expresividad propia del problema. En este trabajo hemos presentado además un algoritmo llamado HSA para el manejo de CSPs no binarios sobre dominios continuos, el cual ha sido aplicado a problemas reales como por ejemplo el análisis ROC [8] donde toda su problemática se reduce a plantear un CSP no binario y continuo donde se requiere obtener la envoltura convexa de todas las soluciones del problema con el objetivo de obtener el volumen que ocupan dichas soluciones.

7. Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto DPI2001-2094-C03-03 del MCYT y el proyecto CTIDIB/2002/112 de la Generalitat Valenciana.

Referencias

[1] N.M. Adams and D.J. Hand. Comparing classifiers when the misallocation costs are uncertain. *Pattern Recognition*, 32:1139–1147, 1999.

[2] F. Bacchus and P. van Beek. On the conversion between non-binary and binary constraint satisfaction problems. *In proceeding of AAAI-98*, pages 311–318, 1998.

[3] C.B. Barber and H.T. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, 1996.

[4] C. Bessière. Non-binary constraints. *In Proc. Principles and Practice of Constraint Programming (CP-99)*, pages 24–27, 1999.

[5] C. Bessière, P. Meseguer, E.C. Freuder, and J. Larrosa. On forward checking for non-binary constraint satisfaction. *In Proc. Principles and Practice of Constraint Programming (CP-99)*, pages 88–102, 1999.

[6] R. Dechter and J. Pearl. Tree clustering for constraint network. *Artificial Intelligence*, 38:353–366, 1989.

[7] B.V. Faltings and E.M. Gelle. Local consistency for ternary numeric constraints. *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 392–400, 1997.

[8] C Ferri, J Hernandez-Orallo, and M.A. Salido. Volume under the ROC surface for multi-class problems. *Machine Learning: ECML 2003, Proceeding of 14th European Conference on Machine Learning (ECML), LNAI 2837*, pages 108–120, 2003.

[9] E.M. Gelle. *On the generation of locally consistent solution spaces in mixed dynamic constraint problems*. PhD thesis, No. 1826, Swiss Federal Institute of Technology in Lausanne, Switzerland, 1998.

[10] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[11] J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

[12] J. Larrosa. *Algorithms and Heuristics for total and partial Constraint Satisfaction*. Phd Dissertation, UPC, Barcelona, 1998.

[13] O. Lhomme. Consistency techniques for numeric CSPs. *In International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 232–238, 1993.

[14] C. Lottaz. Rewriting numeric csps for consistency algorithms. *In Workshop on Non-Binary Constraints at the International Joint Conference on Artificial Intelligence*, pages E:1–E:13, 1999.

[15] A.K. Mackworth. Consistency in network of relations. *Artificial Intelligence*, 8:99–118, 1977.

[16] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[17] D. Mossman and E. Somoza. ROC curves, test accuracy, and the description of diagnostic tests. *J. Neuropsychiatr. Clin. Neurosci.*, 3:330–3., 1991.

[18] C.S. Peirce. Collected papers, vol. iii. *In C. Hartshorne and P. Weiss*, 1933.

[19] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distribution. *Proc. of The Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 43–48, 1997.

[20] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[21] F. Rossi, C. Petrie, and V. Dhar. On the equivalence of constraint satisfaction problems. *In proceeding of European Conference of Artificial Intelligence*, pages 550–556, 1990.

[22] M.A. Salido. *POLYHEDRA, un Modelo para la Resolución de Problemas de Satisfacción de Restricciones n-arias Mediante Hiper-poliedros*. Tesis Doctoral Nº1530, DSIC, Universidad Politécnica de Valencia, Spain, 2002.

[23] M.A. Salido and F. Barber. An incremental and non-binary CSP solver: The Hyperpolyhedron Search Algorithm. *In Proc. of 7th International Conference on Principles and Practice of Constraint Programming (CP-01). LNCS 2239*, pages 799–780, 2001.

[24] M.A. Salido and F. Barber. A Polynomial Algorithm for Continuous Non-binary Disjunctive CSPs: Extended DLRs. *Knowledge Based Systems. Elsevier Science*, 16:277–285, 2003.

[25] M.A. Salido, A. Giret, and F. Barber. Constraint Satisfaction by means of Dynamic Polyhedra. *Operational Research Proceedings 2001. Ed. Springer Verlag*, 1:405–412, 2001.

[26] D. Sam-Haroud. *Constraints Consistency Techniques for Continuous Domains.* PhD thesis, No. 1423, Swiss Federal Institute of Technology in Lausanne, Switzerland, 1995.

[27] D. Sam-Haroud and Faltings B. Global consistency for continuous constraints. *In proceeding of European Conference of Artificial Intelligence (ECAI-94)*, pages 40–50, 1994.

[28] A. Srinivasan. *Note on the Location of Optimal Classifiers in N-dimensional ROC Space.* Technical Report PRG-TR-2-99, Oxford University Computing Laboratory, Oxford, 1999.

[29] K. Stergiou and T. Walsh. Encoding of non-binary constraints satisfaction problems. *In Proc. of the National Conference on Artificial Intelligence (AAAI-99)*, pages 163–168, 1999.

[30] J. Swets, R. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, pages 82–87, 2000.