UNIVERSIDAD POLITÉCNICA DE VALENCIA

Departamento de Sistemas Informáticos y Computación

TESIS DE MÁSTER

Búsqueda de Soluciones Robustas en Problemas de Satisfacción de Restricciones Dinámicos

Autor: Laura Isabel Climent Aunés
Directores: Dr. Miguel A. Salido y Dr. Federico Barber

Grupo de Inteligencia Artificial, Planificación y Scheduling (IA-GPS)
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46020 Valencia, Spain
Noviembre, 2010

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Generalidades

Muchas de las decisiones que tomamos para resolver diversos problemas cotidianos están sometidas a restricciones; de esta manera, decisiones tan cotidianas como fijar una cita, realizar una compra o planificar un viaje dependen de varios aspectos interdependientes e incluso conflictivos en ocasiones.

Cada problema está sujeto a un conjunto de restricciones que debe ser satisfecho para que la decisión tomada sea válida. Además, cuando se encuentra una solución que satisface plenamente unos criterios, puede que no sea tan apropiada para otros, por lo que obtener una única solución (la óptima) no sería suficiente.

En general, muchos problemas de la vida real pueden ser modelados como problemas de satisfacción de restricciones (Constraint Satisfaction Problem - CSP) y ser resueltos mediante técnicas de programación de restricciones. Estos problemas se incluyen en muy diversas áreas, como inteligencia artificial, investigación operativa, bases de datos, sistemas expertos, diagnosis, etc. Algunos ejemplos de tipos de problemas que pueden ser modelados como CSPs son scheduling, planificación, razonamiento temporal, diseño en la ingeniería, problemas de empaquetamiento, criptografía, toma de decisiones, etc. En general, se trata de problemas grandes y complejos, típicamente de complejidad NP.

La programación de restricciones se define como el estudio de sistemas computacionales basados en restricciones. El objetivo de la programación de restricciones es resolver problemas mediante la declaración de restricciones sobre el dominio del problema y consecuentemente encontrar soluciones a instancias de los problemas de dicho dominio que satisfagan todas las res-

tricciones y, en su caso, optimicen unos criterios determinados. Las etapas básicas para la resolución de un problema CSP son dos: modelización y resolución mediante la aplicación de técnicas específicas para CSPs, que incluyen procesos de búsqueda apoyados con métodos heurísticos y procesos inferenciales.

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. Especialmente en la literatura de satisfacción de restricciones, se puede observar como se ha realizado un gran esfuerzo para incrementar la eficiencia de los algoritmos de satisfacción de restricciones: filtrado, aprendizaje y técnicas distribuidas, técnicas de backtracking mejorado, uso de representaciones eficientes y heurísticas, etc. Este esfuerzo se ha traducido en el diseño de herramientas de razonamiento basado en restricciones que se utilizan para resolver numerosos problemas de la vida real [5].

Sin embargo, muchas de estas técnicas asumen que el conjunto de variables, dominios y restricciones del CSP es conocido y fijo. Esta circunstancia produce una importante limitación, puesto que, en situaciones reales, el CSP asociado al problema, cambia debido al entorno, al usuario y otros agentes [35]. Por este motivo, la solución encontrada para un problema puede dejar de ser válida después de que se produzcan cambios en los parámetros del problema.

La mayoría de los investigadores que trabajaban en problemas de satisfacción de restricciones centran su atención en problemas estáticos. Esta tesis de máster surge de la necesidad de obtener modelos y técnicas que permitan manejar CSPs dinámicos (Dynamic Constraint Satisfaction Problem - DynCSP).

Existen diversas técnicas que permiten modelar problemas dinámicos, en los cuales se pueden producir cambios en los parámetros del problema. Varias técnicas de búsqueda han sido propuestas, siendo la mayoría de ellas estrategias correctivas, ya que se aplican después de que se produzcan los cambios en el problema. Sin embargo, existe otra línea de investigación abierta hacia las estrategias proactivas, las cuales se aplican antes de que sucedan los cambios en el problema. Esta característica presenta una importante ventaja sobre las estrategias correctivas, ya que éstas últimas tienen que calcular/reparar la solución cada vez que se produce un cambio en el problema, lo cual implica un elevado coste computacional adicional, sobre todo en casos en los que, dadas las características del problema de la vida real, se producen cambios frecuentes en el problema.

Esta tesis de máster, se encuentra principalmente en el marco de las estrategias proactivas, ya que se basa en la modelización del problema, de

forma que nuestras técnicas son aplicadas antes de que sucedan cambios en el problema y nos permiten encontrar soluciones que tengan más probabilidad de seguir siendo válidas ante futuros cambios en el problema. La principal ventaja que presenta nuestra técnica frente a otras estrategias proactivas es que éstas son menos eficientes que un resolvedor usual de CSP, debido a que incorporan procesos adicionales durante la búsqueda de soluciones que incrementan el coste computacional. Sin embargo, en este trabajo, se realizan modelizaciones de DynCSPs como CSPs ponderados, que pueden ser resueltos por cualquier resolvedor de CSPs ponderados sin incrementar por ello el coste computacional de la resolución.

El objetivo principal de este trabajo es la búsqueda de soluciones que tengan una alta probabilidad de seguir siendo válidas en el caso de que se produzcan cambios en el problema original: estas soluciones se denominan robustas y son inherentes a los DynCSPs. Así pues, la pregunta que nos hacemos a la hora de resolver un DynCSP es: ¿qué solución es más robusta ante las circunstancias de dinamismo que presenta el problema?. Este trabajo trata de dar respuesta a esta cuestión mediante la modelización de los DynCSPs como CSPs ponderados.

1.2. Problemas de Satisfacción de Restricciones

Un problema de satisfacción de restricciones consiste en un conjunto finito de variables, un dominio de valores para cada variable y un conjunto de restricciones que acotan la combinación de valores que las variables pueden tomar. El objetivo de un CSP es seleccionar un valor para cada variable de manera que se satisfagan todas las restricciones del problema.

Los CSPs son, en general, problemas intratables, es decir, no se conocen algoritmos polinómicos para resolver tales problemas. Por ejemplo un problema con 10 variables, y cada variable con 10 posibles valores, tendría un total de diez mil millones de posibilidades diferentes. Los algoritmos completos para llevar a cabo la búsqueda de soluciones en un CSP están basados en técnicas de backtracking. Consideramos algoritmos completos a aquellos que garantizan encontrar una solución si existe, o probar la insatisfacibilidad en caso contrario.

Los algoritmos de backtraking asignan valores a variables, uno a uno, hasta que se encuentre una asignación consistente de todas las variables, o hasta

que se hayan probado sin éxito todos los valores de alguna variable. Cada vez que se le asigna un valor a una variable, los algoritmos de backtracking comprueban si ese valor es compatible con todos los asignados previamente con respecto a las restricciones del problema. Si se cumple esta condición el algoritmo asigna un valor a la siguiente variable. En caso contrario, la última asignación realizada no es válida, y se le asigna el siguiente valor del dominio. En el caso de que se hayan probado todos los valores posibles de la variable, y hayan fallado, el algoritmo retrocede a la variable asignada anteriormente y asigna el siguiente valor de su dominio. De esta manera si llamamos n al número de variables y d a la talla del dominio de todas las variables, entonces el número de posibles combinaciones de valores a variables es d^n. Esto supone una complejidad de $O(d^n)$.

A continuación vamos a presentar un ejemplo de CSP. El ejemplo trata el conocido problema de coloración del mapa; un problema que se puede modelar como un CSP donde los dominios de las variables constan de tres elementos, y el objetivo se centra en encontrar una asignación de un valor a cada variable de manera que se satisfagan las restricciones.

Una vez explicado el ejemplo de CSP, presentaremos diversas variantes de CSPs, algunas de las cuales, serán explicadas con mayor detalle en capítulos posteriores.

1.2.1. Ejemplo de CSP

El problema de coloración del mapa puede ser formulado como un CSP. En este problema, hay un conjunto de colores y queremos colorear cada región del mapa de manera que las regiones adyacentes tengan distintos colores. En la formulación del CSP, hay una variable por cada región del mapa, y el dominio de cada variable es el conjunto de colores disponible. Para cada par de regiones contiguas existe una restricción sobre las variables correspondientes que no permite la asignación de valores idénticos a las variables. Dicho mapa puede ser representado mediante un grafo donde los nodos son las regiones y cada par de regiones adyacentes están unidas por una arista.

En la Figura 1.1 se muestra el ejemplo del problema de coloración del mapa. Seleccionamos cuatro regiones $\{x, y, z, w\}$ para ser coloreadas. Cada región del mapa se corresponde con una variable en el grafo. Si asumimos que cada región puede colorearse con uno de los tres colores, rojo (r), verde (v) y azul (a), entonces cada variable del grafo tiene tres posibles valores.

Las restricciones de este problema expresan que regiones adyacentes tienen que ser coloreadas con diferentes colores. En el grafo que representa el problema, las variables correspondientes a regiones adyacentes están conec-

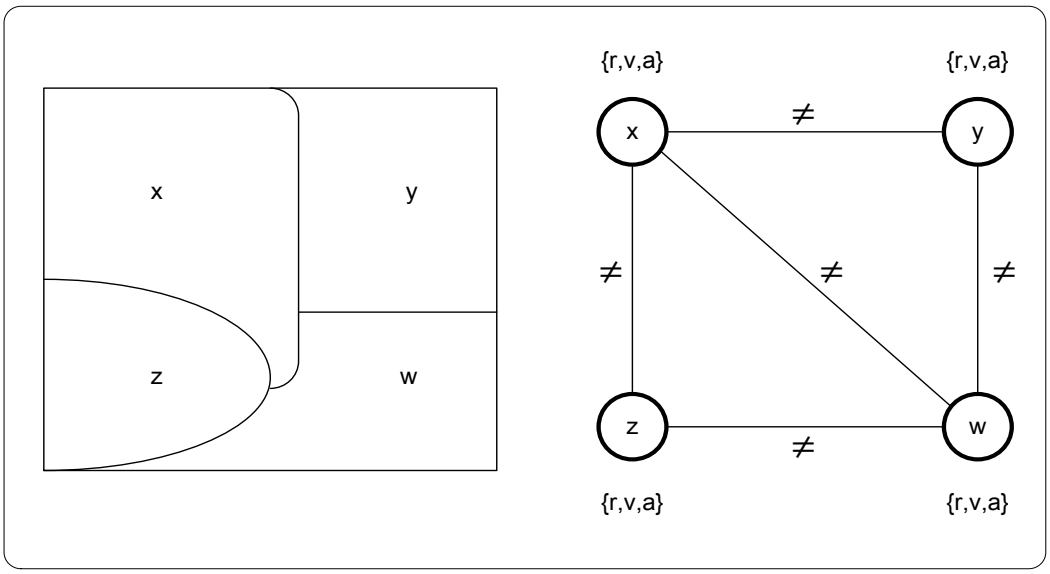

Figura 1.1: Problema de coloración del mapa.

tadas por una arista. Hay cinco restricciones en el problema, es decir, cinco aristas en el grafo. Una solución para el problema es la asignación (x, r), (y, v), (z, v), (w, a). En esta asignación todas las variables adyacentes tienen valores diferentes.

1.2.2. Variantes de CSPs

El modelo clásico de CSP define un modelo estático, con restricciones, variables y dominios fijos e inflexibles. Sin embargo, tal y como se ha comentado anteriormente, en la vida real los problemas suelen ser dinámicos, de manera que se pueden producir cambios en los parámetros del problema. Por este motivo, el modelo clásico y rígido de CSP dificulta la representación de este tipo de problemas.

Además, el modelo clásico de CSP no contempla determinados aspectos como:

- Existen muchas soluciones, pero ¿qué soluciones son mejores que el resto?.
- No existen soluciones, pero ¿algunas asignaciones satisfacen más restricciones que otras?.
- No conocemos las restricciones exactas, pero sí información asociada.

- ¿Estaríamos dispuestos a violar algunas restricciones si a cambio pudiésemos obtener una solución mejor que las demás?.

A continuación vamos a introducir algunas propuestas de variantes de CSP, las cuales han surgido como adaptación del modelo estático de CSP a una gran variedad de problemas.

A. CSPs Dinámicos

Los problemas de satisfacción de restricciones dinámicos (DynCSPs) son capaces de capturar los cambios que se producen en la formulación original de un problema, debido al entorno, al usuario y otros agentes [35].

Los DynCSPs ([9], [2]) representan una secuencia de CSPs estáticos, donde cada uno surge de un cambio producido en el CSP que lo precede, representando nuevos hechos acerca del entorno que se está modelando. Como resultado de este cambio incremental, el conjunto de soluciones del CSP puede potencialmente disminuir (este caso es considerado como una restricción del CSP) o incrementar (este caso es considerado como una relajación del CSP).

Se produce una restricción del CSP cuando se impone una nueva restricción a un subconjunto de las variables del problema (por ejemplo, forzando a una variable a tener un valor determinado), o cuando se añade una nueva variable al CSP. Sin embargo, se produce una relajación del CSP cuando restricciones del problema son eliminadas debido a que han perdido su validez.

En el capítulo 3 se explicarán en extensión los conceptos referentes a CSPs dinámicos.

B. CSPs Flexibles

Existen diversas extensiones del modelo estático de CSP que permiten algunos tipos de flexibilidad. En muchos problemas reales, existen restricciones que podrían ser violadas en soluciones sin causar que tales soluciones fuesen inaceptables. Si estas restricciones se tratan como obligatorias (restricciones duras), tal y como sucede en el modelo estático de CSP, esto podría ocasionar que muchos problemas no pudiesen ser resueltos [1]. Esta es una motivación para ampliar el modelo estático de CSP, de manera que permita algunos casos de flexibilidad como: que las soluciones no tengan porque satisfacer todas las restricciones del CSP o que las soluciones no sólo puedan satisfacer las restricciones completamente, sino que también puedan hacerlo parcialmente.

Los CSPs flexibles tienen en común el uso de un operador que incorpora un nivel de satisfacción de las restricciones. La mejor solución de un CSPs flexible

se encuentra mediante el cálculo de la clasificación global de las asignaciones completas de variables en base a unos determinados criterios que dependen del tipo de CSP flexible [31]. A continuación se explican algunos de los CSPs flexibles más comunes y utilizados:

- Max-CSP: Existen muchos problemas reales sobre-restringidos, ya que no existe ninguna solución que satisfaga todas las restricciones del problema. Para este tipo de problemas, sería interesante encontrar una tupla que fuese la que más respetase el conjunto de restricciones. Los Max-CSPs son CSPs sobre-restringidos para los que la solución es una tupla que satisfaga el mayor número de restricciones del problema ([23], [33]).

- CSPs ponderados: La diferencia principal que introducen los CSPs ponderados sobre el modelo estático de CSP, es que se asocian pesos (o costes) a las asignaciones parciales de cada restricción del problema. De esta forma, se pueden fijar preferencias de satisfacibilidad entre las restricciones del problema mediante los costes asignados a las asignaciones parciales que satisfacen o no cada restricción. El coste total asociado a una tupla, es la suma de todos los costes aplicables. La tupla será consistente si el coste total es inferior a un límite superior establecido. La mejor solución para un CSP ponderado es la solución que tenga el menor coste total asociado posible ([24],[17]). En el capítulo 4 se puede encontrar la definición formal del CSP ponderado.

- CSPs borrosos: En los CSPs borrosos, las restricciones se definen como un par de un conjunto de variables y una relación sobre tales variables. Esta definición nos da la posibilidad de modelar diferentes tipos de incertidumbre en el problema. La relación entre las variables se define mediante una función que indica hasta que punto una tupla satisface la determinada restricción. El rango de valores que puede tomar esta función va desde 0 (la tupla viola completamente la restricción) hasta 1 (la tupla satisface completamente la restricción). Los valores intermedios denotan que la tupla satisface parcialmente la restricción ([1], [38], [25]).

En este trabajo trataremos con CSPs dinámicos debido a que son capaces de capturar las adiciones y eliminaciones de restricciones en el CSP original. La eliminación de restricciones del CSP no es un factor que afecte a la validez de una solución, por ello este tipo de cambios en el problema no son analizados en el proceso de búsqueda de soluciones robustas. Sin embargo, la adición

de restricciones más restrictivas al problema podría invalidar una solución previamente obtenida.

Para conseguir el objetivo de encontrar las soluciones más robustas de un problema dinámico hemos desarrollado dos técnicas que se basan en modelar un DynCSP como un CSP ponderado, consiguiendo de esta forma encontrar soluciones que tengan una elevada probabilidad de permanecer válidas a pesar de que se produzcan cambios en las restricciones del problema original.

1.3. Estructura del Trabajo

Este documento está formado por dos partes principales, las cuales se pueden dividir en seis capítulos. En la primera parte, se presenta el marco teórico del trabajo que incluye el estado del arte de las principales áreas que dan soporte a la propuesta. En el capítulo 2 se detalla en qué consiste la modelización de CSPs, así como distintos algoritmos de búsqueda. En el el capítulo 3 se ofrece una visión general sobre los problemas de satisfacción de restricciones dinámicos, donde se definen los conceptos de robustez y estabilidad, haciendo principal hincapié en sus semejanzas y diferencias.

En la segunda parte del trabajo se presenta el marco de la propuesta donde se detallan las principales aportaciones. En el capítulo 4 proponemos la modelización de DynCSPs como CSPs ponderados, proponiendo técnicas alternativas para: *DynCSPs informados* y *DynCSPs no informados*. En el capítulo 5 se describe el análisis realizado para la evaluación de las técnicas propuestas, tanto en problemas aleatorios como en benchmarks. Finalmente, en el capítulo 6 se presentan las conclusiones, las aportaciones del trabajo de investigación acorde a los objetivos de la misma, las líneas de trabajo futuras y las publicaciones que se han generado durante este trabajo.

Capítulo 2

Problemas de Satisfacción de Restricciones (CSPs)

2.1. Introducción

La resolución de un problema de satisfacción de restricciones (CSP) consta de dos fases diferentes:

- Modelar el problema como un problema de satisfacción de restricciones. La modelización expresa el problema mediante una sintaxis de CSPs, es decir, mediante un conjunto de variables, dominios y restricciones del CSP.

- Resolver el problema de satisfacción de restricciones resultante. Una vez formulado el problema como un CSP, hay diversas técnicas de satisfacción de restricciones para resolver el CSP.

En este capítulo nos centraremos primero en la modelización del problema de satisfacción de restricciones, presentando las definiciones y notación necesarias para modelar un CSP y los distintos niveles en los que se puede modelar un CSP. Para posteriormente explicar cómo el problema modelizado como CSP puede ser manejado con técnicas de búsqueda.

Generalmente la declaración de un problema se suele expresar de muchas maneras diferentes, e incluso en lenguaje natural. Una parte muy importante para la resolución de problemas de la vida real es el modelado del problema en términos de CSPs, es decir, variables, dominios y restricciones.

2.2. Definiciones

En esta sección presentamos los conceptos y objetivos básicos que son necesarios en los problemas de satisfacción de restricciones y que utilizaremos a lo largo de este trabajo.

Definición 2.2.1. *Un problema de satisfacción de restricciones (Constraint Satisfaction Problem - CSP) es una terna* $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ *donde:*

- $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ *es un conjunto de n variables.*
- $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ *es un conjunto de dominios. La* $i-$*ésima componente de D,* D_i *es el dominio que contiene todos los posibles valores que se le pueden asignar a la variable* x_i*.*
- $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ *es un conjunto finito de restricciones. Cada restricción está definida sobre un conjunto de variables* $\{x_1, ..., x_n\}$ *restringiendo los valores que las variables pueden tomar simultáneamente.*

Definición 2.2.2. *Una asignación de variables, también llamada instanciación,* (x, a) *es un par variable-valor que representa la asignación del valor* $'a'$ *a la variable* x*.*

Definición 2.2.3. *Una tupla es una asignación de un conjunto ordenado de variables* $((x_1, a_1), ..., (x_i, a_i))$*, donde cada par ordenado* (x_i, a_i) *asigna el valor* a_i *a la variable* x_i*. Para simplificar la notación, sustituiremos la tupla* $((x_1, a_1), ..., (x_i, a_i))$ *por t.*

El subconjunto $\mathcal{X}_t \subseteq \mathcal{X}$ *está compuesto por las variables que componen la tupla t. Para un subconjunto B de* $\mathcal{X}_t$*, la proyección de t sobre B se denota como* $t \downarrow_B$ *[24].*

Una tupla es localmente consistente si satisface todas las restricciones formadas por las variables de la tupla.

Definición 2.2.4. *Una solución de un CSP es una asignación de valores a todas las variables que componen el CSP, de manera que se satisfagan todas las restricciones. Es decir, una solución es una tupla consistente que contiene todas las variables del problema. Una solución parcial es una tupla consistente que contiene algunas de las variables del problema. Por lo tanto diremos que un problema es consistente, si existe al menos una solución.*

A todo el conjunto de soluciones del CSP se le llama espacio de soluciones. El producto cartesiano de todos los dominios de las variables de un CSP es su espacio de búsqueda. En general, los espacios de soluciones pueden aproximarse o acotarse mediante algoritmos de consistencia. Estos algoritmos utilizan inconsistencias locales entre variables para podar partes del espacio de búsqueda, donde se espera que no haya ninguna solución. En general, los algoritmos que aseguran bajos grados de consistencia sobreestiman el espacio de soluciones pero tienen bajo coste computacional. Los algoritmos que aseguran grados de consistencia más altos proporcionan un espacio de soluciones más ajustado pero tienen más complejidad.

2.3. Representación de Restricciones

En esta sección veremos algunas definiciones sobre restricciones y explicaremos algunas de sus propiedades.

Definición 2.3.1. *La aridad de una restricción es el número de variables que componen dicha restricción. Una restricción unaria es una restricción que consta de una sola variable. Una restricción binaria es una restricción que consta de dos variables. Una restricción ternaria consta de tres variables.*

Una restricción no binaria (o $n-aria$) es una restricción que involucra a un número arbitrario de variables.

Ejemplo. La restricción $x_1 \leq 5$ es una restricción unaria sobre la variable x_1. La restricción $x_4 - x_3 \neq 3$ es una restricción binaria. La restricción $2x_1 - x_2 + 4x_3 \leq 4$ es una restricción ternaria. Por último, un ejemplo de restricciones $n-$aria sería $x_1 + 2x_2 - x_3 + 5x_4 \leq 9$.

Definición 2.3.2. *Una tupla t de una restricción C_i formada por variables $\{x_j, ..., x_k\}$, es un elemento del producto cartesiano $D_j \times .. \times D_k$. Una tupla t que satisface la restricción C_i se le llama tupla permitida o válida. Una tupla t que no satisface la restricción C_i se le llama tupla no permitida o no válida.*

Definición 2.3.3. *Una restricción C_i formada por variables $\{x_j, ..., x_k\}$, es convexa si la línea recta entre cualquier par de puntos $(a_j, ..., a_k)$ y $(b_j, ..., b_k)$ que satisfacen dicha restricción está dentro de la región factible que llamaremos $S(C)$, es decir, dada una restricción C_i, diremos que es convexa si $\forall (a_j, ..., a_k), (b_j, ..., b_k) \in S(C), \exists \alpha \in [0, 1] : (a_j, ..., a_k)\alpha + (b_j, ..., b_k)(1 - \alpha) \in S(C)$.*

Definición 2.3.4. *El grado de restringibilidad de una restricción representa el porcentaje de tuplas posibles que la restricción prohíbe. Las tuplas posibles para una restricción C_i son los elementos del producto cartesiano de los dominios de las variables involucradas en C_i. El grado de restringibilidad de una restricción se define en el intervalo [0,1].*

Ejemplo. Una restricción ternaria en la que cada variable tiene un tamaño de dominio de 10, tiene un conjunto de posibles tuplas de $10^3 = 1000$. Sin embargo, la restricción sólo permite 250 tuplas de todo este conjunto. Es decir, del conjunto de tuplas posibles, 750 no están permitidas, lo que

representa un 75 % del conjunto de tuplas posibles, por lo que el grado de restringibilidad de la restricción es de 0.75.

Una restricción puede definirse *extensionalmente* o *intencionalmente*, siendo ambas representaciones equivalentes:

- Representación intencional: La restricción se representa como una función matemática o lógica.

  Ejemplo. $x_1 \geq 3$ es una restricción intensional unaria.

- Representación extensional: La restricción se representa como un conjunto de tuplas válidas o no válidas.

  Ejemplo. El conjunto de tuplas $\{(3), (4), (5)\}$ es la representación extensional por medio de tuplas válidas, de la restricción intensional $x_1 \geq 3$, considerando un dominio $D_1 : \{0..5\}$ para la variable x_1.

Ejemplo. Consideremos una restricción entre 4 variables x_1, x_2, x_3, x_4, con dominios discretos $\{1, 2\}$, donde la suma entre las variables x_1 y x_2 es menor o igual que la suma entre x_3 y x_4. Esta restricción puede representarse *intencionalmente* mediante la expresión $x_1 + x_2 \leq x_3 + x_4$. Además, esta restricción también puede representarse *extensionalmente* mediante el conjunto de tuplas permitidas $\{(1, 1, 1, 1), (1, 1, 1, 2), (1, 1, 2, 1), (1, 1, 2, 2), (2, 1, 2, 2), (1, 2, 2, 2), (1, 2, 1, 2), (1, 2, 2, 1), (2, 1, 1, 2), (2, 1, 2, 1), (2, 2, 2, 2)\}$, o mediante el conjunto de tuplas no permitidas $\{(1, 2, 1, 1), (2, 1, 1, 1), (2, 2, 1, 1), (2, 2, 1, 2), (2, 2, 2, 1)\}$.

Las restricciones de un CSP pueden ser clasificadas como restricciones duras o blandas:

- Las restricciones duras son aquellas que se tienen que cumplir en todo caso (no pueden ser violadas).
- Las restricciones blandas son aquellas que pueden incumplirse o relajarse (pueden ser violadas), si bien a un cierto coste.

2.4. Algoritmos para la Resolución de CSPs

Las técnicas más usuales que se llevan a cabo para manejar los CSPs se pueden agrupar en tres tipos: Búsqueda sistemática, técnicas inferenciales y técnicas híbridas. Además, existen heurísticas que ayudan a estas técnicas a encontrar soluciones de una manera más eficiente.

2.4.1. Métodos de Búsqueda

Los métodos de búsqueda se centran en explorar el espacio de estados del problema. El espacio de estados del problema es el conjunto de todos los estados alcanzables a partir del estado inicial mediante una secuencia de acciones cualquiera. Estos métodos pueden ser completos, explorando todo el espacio de estados en busca de una solución, o incompletos si solamente exploran una parte del espacio de estados. Los métodos que exploran todo el espacio de búsqueda garantizan encontrar una solución, si existe, o demuestran que el problema no es resoluble. La desventaja de estos algoritmos es que son muy costosos. Los dos métodos completos más usuales son:

- **Generar y Testear(GT):** Este método genera las posibles tuplas de instanciación de todas las variables de forma sistemática y después testea sucesivamente sobre cada instanciación si se satisfacen todas las restricciones del problema. La primera combinación que satisfaga todas las restricciones, será la solución al problema. Mediante este procedimiento, el número de combinaciones generado por este método es el producto cartesiano de la cardinalidad de los dominios de las variables. Esto es un inconveniente, ya que se realizan muchas instanciaciones erróneas de valores a variables que después son rechazadas en la fase de testeo. Por ejemplo, para el caso del problema de las 4-Reinas, se generarían $4^4 = 256$ tuplas a testear. El proceso de generación requeriría $256 * 4 = 1024$ asignaciones de variable.

- **Backtracking Cronológico (BT):** Este método realiza una exploración en profundidad del espacio de búsqueda, instanciando sucesivamente las variables y comprobando ante cada nueva instanciación si las instanciaciones parciales ya realizadas son localmente consistentes. Si es así, sigue con la instanciación de una nueva variable. En caso de conflicto, intenta asignar un nuevo valor a la última variable instanciada, si es posible, y en caso contrario retrocede a la variable asignada inmediatamente anterior.

2.4.2. Técnicas de Inferencia

Los procesos inferenciales en un CSP tienen como objetivo deducir nuevas restricciones, derivadas de las explícitamente conocidas sobre el problema. Concretamente, estas técnicas borran valores inconsistentes de los dominios de las variables o inducen restricciones implícitas entre las variables, obteniendo un nuevo CSP, equivalente al inicial, donde se han hecho explícitas las

nuevas restricciones implícitamente contenidas en el primero. La obtención de estas nuevas restricciones permite:

1. Obtener respuestas a preguntas sobre el problema, relativas a restricciones implícitamente existentes entre las variables o sobre sus dominios.

2. Acotar el espacio de soluciones, al haberse eliminado valores inconsistentes, haciendo más eficientes los procesos de búsqueda.

Las inconsistencias locales son valores individuales o combinación de valores de las variables que no pueden participar en la solución. Por ejemplo, si el valor a de la variable x es incompatible con todos los valores de una variable y pendiente de asignación, ligada a x mediante una restricción, entonces a es inconsistente y no formará parte de ninguna solución del problema. Por lo tanto si forzamos alguna propiedad de consistencia local podemos borrar todos los valores que son inconsistentes con respecto a dicha propiedad.

Los procesos inferenciales están ligados al nivel de consistencia. Un proceso inferencial completo deduciría toda la información contenida en el CSP y va ligado a un nivel de consistencia global. Sin embargo, tiene en general un coste exponencial. Los procesos inferenciales incompletos deducen solo parte de la información contenida en el CSP y van ligados a niveles de consistencia local. En general, su coste es de tipo polinómico, por lo que resultan más interesantes y aplicables. Veamos a continuación distintos niveles de consistencia en un CSP.

- **Consistencia de Nodo (1-consistencia)**

  La consistencia local más simple de todas es la *consistencia de nodo* o *nodo-consistencia*. Forzar este nivel de consistencia nos asegura que todos los valores de una variable satisfacen todas las restricciones unarias sobre esa variable.

  Así, un problema es *nodo-consistente* si y sólo si todas sus variables son nodo-consistentes:

  $$\forall x_i \in X, \forall C_i, \exists a \in D_i : a \text{ satisface } C_i$$

  Ejemplo. Consideremos una variable x en un problema con dominio continuo $[2, 15]$ y la restricción unaria $x \leq 7$. La consistencia de nodo eliminará el intervalo $]7, 15]$ del dominio de x. En la Figura 2.1 mostramos el resultado de aplicar nodo-consistencia a la variable x.

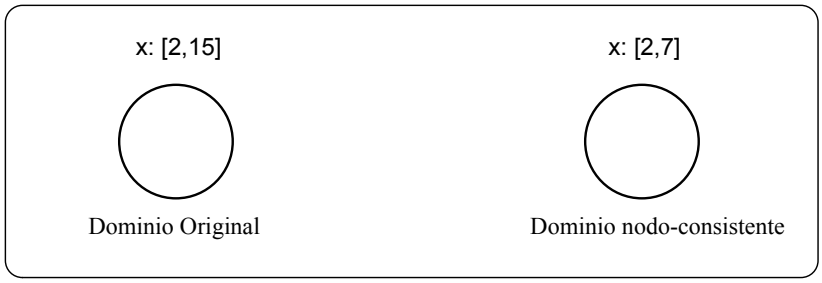

Figura 2.1: Consistencia de nodo, (nodo-consistencia).

- **Consistencia de Arco (2-consistencia)**

  La consistencia local más utilizada es la *consistencia de arco* o *arco-consistencia* [26]. Un problema binario es arco-consistente si para cualquier par de variables restringidas x_i y x_j, para cada valor a en D_i hay al menos un valor b en D_j tal que las asignaciones (x_i, a) y (x_j, b) satisfacen la restricción entre x_i y x_j. Cualquier valor en el dominio D_i de la variable x_i que no es arco-consistente puede ser eliminado de D_i ya que no puede formar parte de ninguna solución. El dominio de una variable es arco-consistente si todos sus valores son arco-consistentes.

  Así, un problema es *arco-consistente* si y sólo si todos sus arcos son arco-consistentes:

  $$\forall\, C_{ij} \in C, \forall a \in D_i, \exists b \in D_j \text{ tal que } b \text{ es un soporte para } a \text{ en } C_{ij}.$$

  Ejemplo. Dada la restricción $C_{ij} = x_i < x_j$ de la Figura 2.2, podemos observar que el arco C_{ij} es consistente, ya que para cada valor $a \in [3, 6]$ hay al menos un valor $b \in [8, 10]$ de manera que se satisface la restricción C_{ij}. Sin embargo si la restricción fuese $C_{ij} = x_i > x_j$ no sería arco-consistente.

- **Consistencia de Caminos**

  La consistencia de caminos [28] (Path-consistency) es un nivel más alto de consistencia local que la arco-consistencia. La consistencia de caminos requiere para cada par de valores a y b de dos variables x_i y x_j, que la asignación de a a x_i y de b a x_j satisfaga la restricción entre x_i y x_j, y que además exista un valor para cada variable a lo largo del camino entre x_i y x_j de forma que todas las restricciones a lo largo del camino

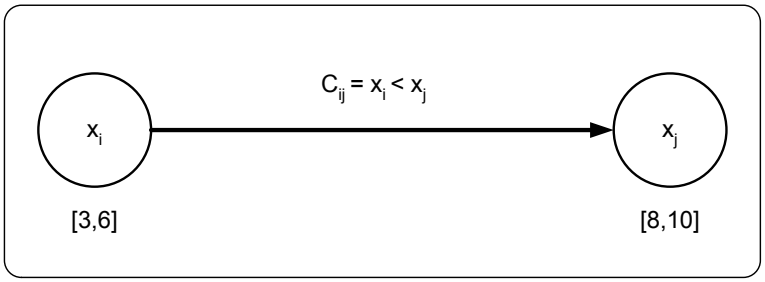

Figura 2.2: Consistencia de arco, (arco-consistencia).

se satisfagan. Montanari demostró que un CSP satisface la consistencia de caminos si y sólo si todos los caminos de longitud dos cumplen la consistencia de caminos [28]. Cuando un problema satisface la consistencia de caminos y además es nodo-consistente y arco-consistente se dice que satisface fuertemente la consistencia de caminos (strongly path-consistent).

El estudio de la consistencia de caminos no se utiliza con mucha frecuencia debido a su alto coste temporal y espacial, y también debido al hecho de que puede modificar el grafo de restricciones añadiendo nuevas restricciones binarias. Sin embargo es un nivel de consistencia importante en CSPs con ciertas propiedades como los CSPs temporales [11].

2.4.3. Técnicas Híbridas

En la Sección 2.4.1 se han visto técnicas de búsqueda de soluciones en un CSP que, en general, resultan exponenciales con el tamaño del problema. Por otra parte, en la Sección 2.4.2 se han visto técnicas inferenciales que eliminan valores inconsistentes de los dominios de las variables o inducen nuevas restricciones implícitas entre las variables restringiendo las previas, de forma que obtienen un nuevo CSP, más restringido y equivalente al inicial, que permite acotar el espacio de búsqueda. Por ello, las técnicas inferenciales se usan como etapas de preproceso donde se detectan y se eliminan inconsistencias locales antes de empezar la búsqueda con el fin de reducir el árbol de búsqueda. Dependiendo del nivel de consistencia del preproceso, se acotará más o menos el espacio de búsqueda, a costa de un mayor o menor esfuerzo computacional en el proceso inferencial previo.

Por otra parte, las técnicas inferenciales pueden incluirse en el propio proceso de búsqueda, dando lugar a los algoritmos de búsqueda híbridos que

analizamos en esta sección. La complejidad exponencial para resolver un CSP dado está principalmente relacionada con el tamaño de los dominios de sus variables.

- **Algoritmos Look-Backward:** Los algoritmos Look-Backward tratan de explotar la información del problema para comportarse más eficientemente en las situaciones sin salida. Al igual que el backtracking cronológico, los algoritmos Look-Backward llevan a cabo la comprobación de la consistencia hacia atrás, es decir, entre la variable actualmente instanciada y las pasadas ya instanciadas.

- **Algoritmos Look-Ahead:** Los algoritmos Look-Ahead hacen una comprobación inferencial hacia adelante en cada instanciación, integrando un proceso inferencial durante el propio proceso de búsqueda, por lo que también se denominan técnicas híbridas. Esto también se conoce como la propagación de los efectos de cada nueva instanciación al resto de la red. Ello permite: (i) acotar las restricciones y dominios de las variables futuras a instanciar, limitando el espacio de búsqueda pendiente, y (ii) encontrar las inconsistencias antes de que aparezcan, en el caso de que las instanciaciones parciales efectuadas se descubran inconsistentes con el resto de variables pendientes. En definitiva, intentan descubrir si la actual asignación localmente consistente de las k variables puede ser extendida a una solución global, provocando en caso contrario un punto de backtracking.

2.4.4. Heurísticas

Un algoritmo de búsqueda para la satisfacción de restricciones requiere el orden en el cual se van a estudiar las variables, así como el orden en el que se van a estudiar los valores de cada una de las variables. Seleccionar el orden correcto de las variables y de los valores puede mejorar notablemente la eficiencia de resolución. Las heurísticas de ordenación de variables y de valores juegan un papel importante en la resolución de CSPs. A continuación se explican algunas de las heurísticas más conocidas.

A. Ordenación de Variables

Experimentos y análisis de muchos investigadores han demostrado que el orden en el cual las variables son asignadas durante la búsqueda puede tener una impacto significativo en el tamaño del espacio de búsqueda explorado.

Las heurísticas de *ordenación de variables estáticas* generan un orden fijo de las variables antes de iniciar la búsqueda, basado en información global derivada de la topología del grafo de restricciones original que representa el CSP.

En la literatura se han propuesto varias heurísticas de ordenación de variables estáticas, siendo las más comunes:

- La heurística *minimum width* (MW) [13] considera la *anchura* de la variable x como el número de variables que están antes de x, de acuerdo a un orden dado, y que son adyacentes a x. Las variables se ordenan desde la última hasta la primera en anchura decreciente. Esto significa que las variables que están al principio de la ordenación son las más restringidas y las variables que están al final de la ordenación son las menos restringidas. Asignando las variables más restringidas al principio, las situaciones sin salida se pueden identificar antes y además se reduce el número de vueltas atrás. Esta heurística es útil especialmente para CSPs donde el grado de los nodos en el grafo de restricciones varia significativamente.

- La heurística *maximun degree* (MD) [10] ordena las variables en un orden decreciente de su grado en el grafo de restricciones. El *grado* de un nodo se define como el número de nodos que son adyacentes a él. Esta heurística también tiene como objetivo encontrar un orden de anchura mínima, aunque no lo garantiza.

- La heurística *maximun cardinality* (MC) [30] selecciona la primera variable arbitrariamente y después en cada paso, selecciona la variable que es adyacente al conjunto más grande de las variables ya seleccionadas. La heurística MC puede generalizarse a CSPs no binarios de una forma directa.

Las heurísticas de *ordenación de variables dinámicas* pueden cambiar el orden de las variables dinámicamente basándose en información local que se genera durante la búsqueda. Generalmente las heurísticas de ordenación de variables tratan de seleccionar lo antes posible las variables que más restringen a las demás. La idea principal es tratar de asignar lo antes posible las variables más restringidas y de esa manera identificar las situaciones sin salida lo antes posible y así reducir el número de vueltas atrás.

En la literatura se han propuesto varias heurísticas de ordenación de variables dinámicas, siendo las más comunes:

- La heurística de ordenación de variables dinámicas más común se basa en el principio de *primer fallo* (FF) [18] que sugiere que *para tener éxito*

deberíamos intentar primero donde sea más probable que falle. De esta manera las situaciones sin salida pueden identificarse antes y además se ahorra espacio de búsqueda. De acuerdo con el principio de FF, en cada paso, seleccionaríamos la variable más restringida. La heurística FF también conocida como heurística *minimum remaining values* (MRV), trata de hacer lo mismo seleccionando la variable con el dominio más pequeño. Esto se basa en la idea de que si una variable tiene pocos valores, entonces es más difícil encontrar un valor consistente. En cada etapa de la búsqueda, la próxima variable a asignarse es la variable con el dominio más pequeño.

- En [22] se introduce una heurística que trata de seleccionar la variable más restringida evaluando cuán restringidos están los valores de cada variable. Es decir, para cada variable x no instanciada, se evalúan los valores de x con respecto al número de valores de las futuras variables que no son compatibles. Entonces estas evaluaciones se combinan para producir una estimación de cuán restringida está x. La variable más restringida se selecciona.

- Geelen propuso una heurística similar basada en el conteo del número de valores soporte para cada valor de una variable no instanciada [15]. Geelen llamó *promesa* de un valor a al producto de los valores que lo soportan en todas las variables futuras. De esta manera, esta heurística mide la promesa de los valores para cada variable no asignada y selecciona la variable cuya suma de promesas de sus valores es mínima. La heurística trata así de minimizar el número total de valores que pueden asignarse a todas las variables futuras de forma que no se viole ninguna restricción de la variable seleccionada. Un efecto lateral de la heurística de ordenación de variables de Geelen es que se asegura la arco-consistencia en cada etapa de la búsqueda, pero por el contrario es una heurística costosa de calcular. Su complejidad temporal para una asignación de un valor a una variable es $O(n^2d^2)$ donde n es el número de variables y d es la talla del dominio más grande.

B. Ordenación de Valores

La idea básica que hay detrás de las heurísticas de ordenación de valores es seleccionar el valor de la variable actual que más probabilidad tenga de llevarnos a una solución, es decir identificar la rama del árbol de búsqueda que sea más probable que obtenga una solución. La mayoría de las heurísticas propuestas tratan de seleccionar el valor menos restringido de la variable

actual, es decir, el valor que menos reduce el número de valores útiles para las futuras variables.

Una de las heurísticas de ordenación de valores más conocidas es la heurística *min-conflicts* [27]. Básicamente, esta heurística ordena los valores de acuerdo a los conflictos en los que éstos están involucrados con las variables no instanciadas. Cada valor a de la variable x_i se asocia con el número total de tuplas que son incompatibles con a en las restricciones en las que está involucrada la variable x_i. De manera que se selecciona el valor con la menor suma. En [22] Keng y Yun proponen una variación de la idea anterior. De acuerdo a su heurística, cuando se cuenta el número de valores incompatibles para una futura variable x_k, éste se divide por la talla del dominio de x_k. Esto da el porcentaje de los valores útiles que pierde x_k debido al valor a que actualmente estamos examinando. De nuevo los porcentajes se añaden para todas las variables futuras y se selecciona el valor más bajo que se obtiene en todas las sumas.

Geelen propuso una heurística de ordenación de valores a la cual llamó *promise* [15]. Para cada valor a de la variable x contamos el número de valores que soporta a en cada variable adyacente futura, y toma el producto de las cantidades contadas. Este producto se llama la promesa de un valor. De esta manera se selecciona el valor con la máxima promesa. Usando el producto en vez de la suma de los valores soporte, la heurística de Geelen trata de seleccionar el valor que deja un mayor número de soluciones posibles después de que este valor se haya asignado a la variable actual. La promesa de cada valor representa una cota superior del número de soluciones diferentes que pueden existir después de que el valor se asigne a la variable.

En [14] se describen tres heurísticas de ordenación de valores dinámicos inspirados por la intuición de que un subproblema es más probable que tenga solución si no tiene variables que tengan un sólo valor en su dominio.

- La primera heurística, llamada heurística *max-domain-size* selecciona el valor de la variable actual que crea el máximo dominio mínimo en las variables futuras.

- La segunda heurística, llamada *weighted-max-domain-size* es una mejora de la primera. Esta heurística especifica una manera de romper empates basada en el número de futuras variables que tiene una talla de dominio dado. Por ejemplo, si un valor a_i deja cinco variables futuras con dominios de dos elementos, y otro valor a_j deja siete variables futuras también con dominios de dos elementos, en este caso se selecciona el valor a_i.

- La tercera heurística, llamada *point-domain-size*, que asigna un peso

(unidades) a cada valor de la variable actual dependiendo del número de variables futuras que se quedan con ciertas tallas de dominios. Por ejemplo, para cada variable futura que se queda con un dominio de talla uno debido a la variable a_i, se añaden 8 unidades al peso de a_i. De esta manera se selecciona el valor con el menor peso.

Sin embargo, estas heurísticas parecen ser muy a medida y los resultados experimentales presentados en [14] muestran que la heurística *min-conflict* supera a estas tres últimas.

Capítulo 3

Problemas de Satisfacción de Restricciones Dinámicos (DynCSPs)

3.1. Introducción

Muchos problemas de la vida real pueden ser modelados como problemas de satisfacción de restricciones y ser resueltos utilizando técnicas de programación de restricciones. Se ha realizado mucho esfuerzo entorno al incremento de la eficiencia de los algoritmos de satisfacción de restricciones: filtrado, aprendizaje y técnicas distribuidas, backtracking mejorado, el uso de representaciones eficaces, heurísticas, etc. Este esfuerzo ha desencadenado en el diseño de herramientas de razonamiento con restricciones, las cuales se utilizan para resolver numerosos problemas reales [5].

Sin embargo, muchas de estas técnicas asumen que el conjunto de variables, dominios y restricciones que componen el CSP es conocido y fijo. Esta circunstancia produce una importante limitación, puesto que, en situaciones reales, el problema cambia debido al entorno, al usuario y otros agentes [35]. De esta forma, la solución encontrada para un problema puede dejar de ser válida después de que estos cambios en el problema sucedan.

Este trabajo surge de la necesidad de obtener modelos y técnicas que permitan manejar CSPs dinámicos, siendo el objetivo principal, encontrar soluciones robustas para tales problemas.

En este capítulo serán definidos conceptos referentes a los CSPs dinámicos, así como una revisión de las diferentes técnicas propuestas para solucionar estos problemas.

3.2. Estado del Arte

El mundo real es dinámico en su naturaleza, por lo que se deben tener en cuenta nuevas técnicas que tratan de modelar problemas dinámicos.

Definición 3.2.1. *Un problema de satisfacción de restricciones dinámico (Dynamic Constraint Satisfaction Problem - DynCSP) es una secuencia de CSPs estáticos donde cada uno surge de un cambio producido en el CSP que lo precede, representando nuevos hechos acerca del entorno que se está modelando [9].*

Como resultado de este cambio incremental, el conjunto de soluciones del CSP puede potencialmente disminuir (este caso es considerado como una restricción del CSP) o incrementar (este caso es considerado como una relajación del CSP).

Se produce una restricción del CSP cuando se impone una nueva restricción a un subconjunto de las variables del problema (por ejemplo, forzando a una variable a tener un valor determinado), o cuando se añade una nueva variable al CSP. Sin embargo, se produce una relajación del CSP cuando restricciones del problema son eliminadas debido a que han perdido su validez.

Los DynCSPs son una extensión de los CSPs estáticos que al ser capaces de modelar la adición y eliminación de restricciones, son más apropiados para el manejo de problemas dinámicos del mundo real. De hecho, es fácil ver que todas las posibles modificaciones en las restricciones o dominios de un CSP se pueden expresar en términos de la adición o eliminación de restricciones [35].

A lo largo del tiempo, se han propuesto varias técnicas para resolver los DynCSPs. La mayoría de estas técnicas son estrategias correctivas, ya que se aplican después de que se produzcan los cambios en el problema. Por lo tanto, estas estrategias sólo se aplican si los cambios que se dan en el CSP tienen como consecuencia que la solución encontrada originalmente (antes de que sucediesen los cambios) deje de ser válida. Las estrategias correctivas están orientadas a encontrar una nueva solución que sea lo más similar posible a la solución original o bien intentan reparar la solución original, la cual ha dejado de ser válida, tratando de hacer los mínimos cambios posibles. Las estrategias correctivas pueden ser clasificadas en dos grupos [4]:

- *Métodos heurísticos*, que utilizan información sobre las partes afectadas

de cualquier asignación previa consistente como heurística para solucionar el nuevo problema [37].

- *Métodos de reparación local*, que consisten en utilizar cualquier asignación previa consistente y repararla, utilizando una secuencia de modificaciones locales ([12], [35]).

Algunas estrategias correctivas existentes en la literatura, centran su atención en el análisis de problemas de scheduling dinámicos ([12], [35]). Estas técnicas, han sido diseñadas para reconfigurar mínimamente los *schedules* en respuesta a un entorno dinámico. En estos entornos, factores externos han provocado que el *schedule* existente deje de ser válido, puede ser que debido a la retirada de recursos, la llegada de nuevos recursos o debido a cambios en el conjunto de las actividades programadas. Estas técnicas se encargan de buscar un nuevo *schedule* que difiera mínimamente del original, el cual ha dejado de ser válido tras los cambios que se han producido en el problema. Para ello, en primer lugar modelan el problema como un DynCSP y posteriormente aplican los métodos heurísticos o de reparación local. A continuación se explican algunos ejemplos de este tipo de técnicas:

- Un ejemplo de técnica aplicada a problemas de *scheduling* dinámicos se basa en la reducción de la *contention* en las restricciones [12]. Se dice que una restricción está sometida a *contention* cuando algunas combinaciones de los valores de los dominios de las variables que la componen, podrían violar la restricción. Para algunas restricciones, se puede medir la *contention*, de manera que la búsqueda puede ser dirigida hacia regiones donde la *contention* es reducida. Una vez se ha completado la fase de viabilidad de los recursos, se puede pasar el control a fase de optimización temporal del *schedule*.

- Un ejemplo de técnica aplicada a problemas de *scheduling* dinámicos en los cuales se introducen nuevas actividades [35], se basa en la idea de es que es posible introducir una nueva actividad t si existe para t una ubicación tal que todas las actividades cuya ubicación es incompatible con la ubicación de t se pueden eliminar. De manera que se puedan introducir otras actividades sin que sea necesario modificar la ubicación de la nueva actividad t.

Además, existen otras estrategias correctivas que centran su atención en los DynCSPs a nivel general, de manera que simulan posibles cambios en problemas originales, intentando encontrar soluciones que difieran lo mínimo

posible de las soluciones previas (soluciones encontradas antes de que los cambios sucediesen) [37].

Existe otra vertiente de investigación abierta hacia la búsqueda de soluciones que tengan más probabilidad de seguir siendo válidas ante cambios que temporalmente alteran el conjunto de asignaciones válidas. Este tipo de estrategias se llaman estrategias proactivas y surgen de la necesidad de resolver *DynCSPs recurrentes*. Estos problemas son llamados de tal forma para distinguirlos de los *DynCSPs permanentes*. En los *DynCSPs recurrentes*, las alteraciones en el problema son temporales, es decir, los cambios pueden ocurrir repetidamente y diferentes cambios pueden ocurrir con diferentes frecuencias. En estos casos, debería ser posible ir más allá de las estrategias correctivas, estudiadas en trabajos anteriores, hacía estrategias más proactivas.

Una estrategia proactiva que ha sido desarrollada para resolver una clase restringida de *DynCSPs recurrentes* en los que los valores podrían ser perdidos temporalmente; consiste básicamente en penalizar los valores que han dejado de ser válidos debido a los cambios en el problema. Así, incluso si estos valores pueden ser utilizados posteriormente, el algoritmo tratará de encontrar soluciones que no los incluyan. Esta idea se incorpora a una técnica de *hill-climbing*, donde las penalizaciones son utilizadas por *min-conflicts* para realizar la selección de valores para las variables [36].

Además de las estrategias correctivas y proactivas, existen algunas técnicas que pueden considerarse como estrategias híbridas, por el hecho de poseer características de ambas estrategias. Las técnicas de búsqueda de super-soluciones ([20], [19]) se pueden considerar como híbridas debido a que realizan una tarea de búsqueda antes de que se produzcan los cambios en el problema, para ser capaz de reparar la solución invalida después de que ocurran los cambios. Informalmente, una solución es una super-solución si es posible reparar la solución con sólo unos pocos cambios cuando un pequeño número de variables pierden sus valores. Una definición más formal es: una solución S es una *(a,b)*-super-solución si la pérdida de los valores en a variables en S a lo sumo, puede ser reparada mediante la asignación de otros valores a estas variables, y modificación de las asignaciones de b otras variables como máximo. Decidir si un CSP tiene una *(a,b)*-super-solución es NP-completo para cualquier a fijada. Principalmente, los trabajos desarrollados en esta área, se centran en la búsqueda de (1,0)-super-soluciones.

Ejemplo. Consideremos el siguiente CSP:

$x_0, x_1 \in \{1, 2, 3\}$
$C_1 : x_0 \leq x_1$

- La solución $(x_0 = 1, x_1 = 1)$ no es una (1,0)-super-solución. Debido a

que si la variable x_0 pierde su valor 1, no es posible encontrar otro valor para la variable que sea consistente con x_1, ya que $(x_0 = 2, x_1 = 1)$ y $(x_0 = 3, x_1 = 1)$ no son soluciones del problema.

- La solución $(x_0 = 1, x_1 = 2)$ es una (1,0)-super-solución. Debido a que si cualquier variable pierde su valor es posible encontrar al menos otro valor para que sea compatible con la otra variable. Si x_0 pierde el valor 1, se le puede asignar el valor de 2, ya que $(x_0 = 2, x_1 = 2)$ es solución del problema. Si x_1 pierde el valor 2, se le puede asignar el valor de 1 o 3, ya que $(x_0 = 1, x_1 = 1)$ y $(x_0 = 1, x_1 = 3)$ son soluciones del problema.

Para encontrar super-soluciones, estas técnicas desarrollan algoritmos de búsqueda basados en super consistencia (adaptaciones de las técnicas de arco-consistencia para las super-soluciones). Sin embargo, encontrar (1,0)-super-soluciones puede ser muy difícil porque (1) la existencia de una variable *backbone* (una variable que toma el mismo valor en todas las soluciones) garantiza que no existan (1,0)-super-soluciones; y (2) los experimentos muestran que es bastante raro encontrar (1,0)-super-soluciones donde todas las variables puedan ser reparadas. Por ello, también se desarrollan algoritmos *branch and bound* que encuentran una solución que sea lo más parecida posible a una (1,0)-super-solución. Es decir, que el número de variables reparables sea el máximo posible. Dada una solución S, una variable es reparable si existe al menos un valor en su dominio diferente al asignado en S, que sea compatible con todos los otros valores en S.

Esta tesis de máster, se enmarca en las estrategias proactivas, ya que las técnicas desarrolladas en este trabajo se basan en la modelización de los DynCSPs como CSPs ponderados, así que las técnicas son aplicadas antes de que los cambios en el problema sucedan. Estas técnicas de modelado nos permiten encontrar soluciones que tienen más probabilidad de seguir siendo válidas ante futuros cambios en el problema.

La principal ventaja que presenta nuestra técnica frente a las estrategias proactivas es que las técnicas proactivas son menos eficientes que un resolvedor usual de CSP, debido a que incorporan procesos adicionales durante la búsqueda de soluciones que incrementan el coste computacional. Sin embargo, en este trabajo, se realizan modelizaciones de DynCSPs, que pueden ser resueltos por cualquier resolvedor de CSPs ponderados sin incrementar por ello el coste computacional de la resolución.

La ventaja principal que presenta nuestra técnica frente a las estrategias correctivas es que las técnicas correctivas tienen que calcular una nueva solución o reparar la solución original cada vez que se produce un cambio en

el problema, mientras que las técnicas desarrolladas en este trabajo buscan soluciones robustas ante futuros cambios en el problema, por lo que estas soluciones tendrán una alta probabilidad de seguir siendo válidas después de que se produzcan los futuros cambios. Nosotros consideramos que es muy importante evitar la pérdida de las soluciones calculadas siempre que sea posible, evitando de la misma forma, que aumente el coste computacional.

3.3. Estabilidad y Robustez

En la literatura no existen conceptos claros y consensuados sobre la estabilidad y robustez de una solución en un CSP. Algunos investigadores hablan acerca de estabilidad y otros acerca de robustez. ¿Cuál es la diferencia entre estabilidad y robustez? Es la primera pregunta que viene a la mente, especialmente, a los investigadores que trabajan con modelos cuantitativos o teorías matemáticas [21]. En primer lugar, analizaremos las diferencias de ambos conceptos a nivel general, para analizar posteriormente los conceptos de robustez y estabilidad de soluciones de CSPs.

Los conceptos de robustez y estabilidad surgen a raíz de la existencia de sistemas dinámicos en los cuales es obligado centrarse en las perturbaciones debido a que representan cambios en la composición o la topología del sistema. Se dice que una solución de un sistema dinámico es estable si pequeñas perturbaciones en la solución resultan en una nueva solución que es "cercana" a la solución original. Sin embargo, el concepto de robustez es más amplio que el concepto de estabilidad. La robustez en un sistema dinámico está asociada a la medida de la característica de persistencia ante las perturbaciones del sistema [21].

Una vez que se han analizado los conceptos de estabilidad y robustez a nivel general, estos pueden ser especificados para soluciones de CSPs. La estabilidad de soluciones de DynCSPs ha sido definida en la literatura, mientras que la de robustez de soluciones de DynCSPs se propone en este trabajo.

Definición 3.3.1. *La estabilidad de una solución es la capacidad de una solución de compartir tantos valores como sea posible con una nueva solución si sucede un cambio [19]. Se mide en términos de similitud de la nueva solución con la solución inicial.*

Definición 3.3.2. *La robustez de una solución es la medida de la persistencia de la solución ante modificaciones en el problema original. Por lo tanto, una solución de un DynCSP es robusta si tiene una alta probabilidad de permanecer válida ante futuros cambios en el problema.*

Posteriormente, en el capítulo 5 se presentarán nuestras propuestas para la medida de la robustez en soluciones para problemas de satisfacción de restricciones.

3.4. DynCSPs Informados y No Informados

En este trabajo vamos a tratar DynCSPs en los que los únicos cambios que se pueden producir están asociados a las restricciones del problema. Además, consideramos que en el mundo real, suelen existir muchos problemas que tienen información asociada acerca del dinamismo de las restricciones, mientras que también los hay, que no tienen dicha información asociada.

En este trabajo introducimos dos alternativas relacionadas con los problemas de satisfacción de restricciones dinámicos: *DynCSPs informados* y *DynCSPs no informados*. De este modo, clasificaremos los DynCSPs como *informados o no informados* dependiendo de si se conoce o no, información acerca del dinamismo de las restricciones del problema. Esta información nos aporta un conocimiento de cómo podrían cambiar las restricciones del problema tras el paso del tiempo y de lo significativos que serían tales cambios. Sin embargo, si no poseemos tal información, no tenemos ningún conocimiento de cómo pueden cambiar las restricciones.

Definición 3.4.1. *Un problema de satisfacción de restricciones dinámico e informado (DynCSP informado) es un problema de satisfacción de restricciones dinámico con información adicional asociada a cada restricción. Esta información está relacionada con la probabilidad y magnitud de cambio de cada restricción.*

Definimos dos funciones que modelan el dinamismo de las restricciones [8]:

- *$p(C_i)$: Es la función de probabilidad de dinamismo. Cada restricción C_i tiene una probabilidad $p(C_i) \in [0, 1[$ que mide la probabilidad de cambio de esta restricción. El valor mínimo que la función puede tomar ($p(C_i) = 0$) significa que la restricción es estática, es decir, no existe posibilidad alguna de que cambie. El valor máximo ($p(C_i) \approx 1$) significa que la restricción es muy dinámica, es decir, que la probabilidad de cambie es muy alta.*

- *$d(C_i)$: Esta función mide la magnitud de cambio de una restricción dinámica, siendo $d(C_i) \in]0, 1[$. Es decir, mide el porcentaje de tuplas válidas que pasarán a ser no válidas después de que se produzca un cambio en la restricción. Un valor de $d(C_i) \approx 0$ significa que la restricción C_i prácticamente no cambiará, por lo que casi todas las tuplas válidas permanecerán siendo válidas después de que la restricción cambie. Sin embargo, $d(C_i) \approx 1$ significa que la mayoría de las tuplas válidas pasarán a ser no válidas después de que se produzca el cambio en la restricción. Esta función no está definida para restricciones estáticas (restricciones cuyo $p(C_i)$=0).*

Ejemplo. A continuación veremos un ejemplo de un *DynCSP informado* formado por una variable x_0 con dominio $D_0 : \{0..5\}$. Cada restricción del *DynCSP informado* es etiquetada con dos números reales. El primer número entre los paréntesis es $p(C_i)$ y el segundo $d(C_i)$:

- $C_1(0.3, 0.2) : x_0 \leq 4$
- $C_2(0.8, 0.4) : x_0 \geq 1$

El conjunto de tuplas válidas para la restricción C_1 es: $\{(x_0 = 0), (x_0 = 1), (x_0 = 2), (x_0 = 3), (x_0 = 4)\}$. El conjunto de tuplas válidas para la restricción C_2 es: $\{(x_0 = 1), (x_0 = 2), (x_0 = 3), (x_0 = 4), (x_0 = 5)\}$.

Las funciones de dinamismo asociadas a C_1 nos indican que $p(C_1) = 0.3$ y $d(C_1) = 0.2$. Es decir, que existe un 30 % de probabilidad de que un 20 % de tuplas válidas de la restricción C_1 dejen de ser válidas debido al dinamismo del problema. Dado que la restricción C_1 tiene 5 tuplas válidas asociadas, el 20 % de 5 tuplas es 1 tupla. Por ello, existe un 30 % de probabilidad de que 1 tupla válida de la restricción C_1 deje de ser válida.

Para la restricción C_2, las funciones de dinamismo son: $p(C_2) = 0.8$ y $d(C_2) = 0.4$. Es decir, que existe un 80 % de probabilidad de que un 40 % de tuplas válidas de la restricción C_2 dejen de ser válidas. La restricción C_2 también tiene 5 tuplas válidas asociadas, el 40 % de 5 tuplas es 2 tuplas. Por ello, existe un 80 % de probabilidad de que dos tuplas válidas de la restricción C_2 dejen de ser válidas.

Definición 3.4.2. *Un problema de satisfacción de restricciones dinámico y no informado (DynCSP no informado) es un problema de satisfacción de restricciones dinámico que no tiene información adicional asociada a cada restricción. Por lo tanto no se conoce ningún dato acerca de la probabilidad y cantidad de cambio de cada restricción.*

En este trabajo se han desarrollado dos técnicas distintas, cada una asociada a un tipo de DynCSP. En el siguiente capítulo (véase capítulo 4), se introducirán ambas técnicas, siendo la primera aplicada a los *DynCSPs informados* y la segunda a los *DynCSPs no informados.*

Capítulo 4

Modelización de DynCSPs como CSPs Ponderados (WCSPs)

4.1. Introducción

En este trabajo, centramos nuestra atención en la búsqueda de soluciones robustas para DynCSPs, las cuales tienen elevadas probabilidades de permanecer válidas a pesar de que se produzcan cambios en las restricciones del problema.

Tal y como se ha comentado en el capítulo 3, el DynCSP captura las adiciones y eliminaciones de restricciones sobre el CSP original. La eliminación de restricciones de un CSP no invalida una solución, por lo que este tipo de cambios no se contemplan en el proceso de búsqueda de soluciones robustas. Sin embargo, la adición de restricciones más restrictivas podría invalidar una solución.

En este trabajo, vamos a considerar posibles modificaciones más restrictivas de las restricciones del problema original. Este tipo de modificación en la restricción, se puede expresar como la eliminación de la restricción original y la adición de la nueva restricción más restrictiva; debido a que la restricción original pasa a ser redundante. Cuanto mayor sea el número de nuevas restricciones modificadas que satisface la solución encontrada, más robusta será la solución, ya que tiene más probabilidad de continuar siendo válida ante más cambios en las restricciones del problema.

En la Figura 4.1 [8] se observa un ejemplo de un CSP al que se le han añadido dos modificaciones más restrictivas (C_{11} y C_{12}) de la restricción original C_1. Estas nuevas restricciones modificadas son consideradas restricciones blandas. Las soluciones que satisfacen C_{11} y C_{12} son consideradas más

robustas que aquellas soluciones que no las satisfacen.

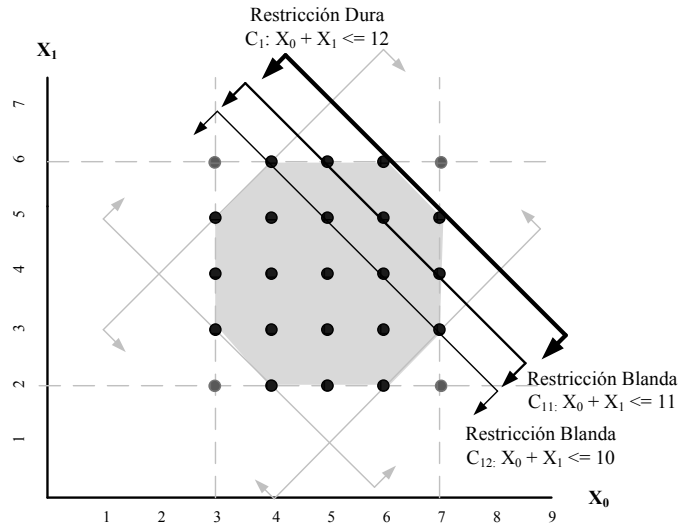

Figura 4.1: Modificaciones más restrictivas de una restricción original [8].

En trabajos previos, se desarrollaron técnicas de creación de restricciones *pseudo-random* [6]. Sin embargo, posteriores trabajos de generación de nuevas restricciones modificadas ([7], [32]) resultaron ser más apropiadas para la búsqueda de soluciones robustas para DynCSPs.

A lo largo de este capítulo se explicará el proceso de modelización de DynCSPs a CSPs ponderados. Mediante tal modelización, pretendemos conseguir el objetivo de encontrar soluciones robustas para los DynCSPs. Para ello, se definirán formalmente los CSPs ponderados, así como el formato utilizado para representarlos. Además, se introducirán dos técnicas diferentes para los dos tipos de DynCSP: DynCSPs informados y DynCSPs no informados.

4.2. Definiciones

Los problemas de satisfacción de restricciones ponderados son una extensión de los problemas de satisfacción de restricciones estándar. La diferencia principal que introducen respecto al modelo original de CSP, es que se asocian pesos (o costes) a las tuplas de cada restricción del problema.

Un problema de satisfacción de restricciones ponderado es una clase específica de los CSPs valuados [34].

Definición 4.2.1. *Un problema de satisfacción de restricciones ponderado (Weighted Constraint Satisfaction Problem - WCSP), se define como* $P = \langle \mathcal{X}, \mathcal{D}, S(k), \mathcal{C} \rangle$ *[24] donde:*

- $\mathcal{X}$ *y* $\mathcal{D}$ *son las variables y dominios respectivamente, tal y como en un CSP estándar.*
- $S(k)$ *es la estructura de valuación, donde* $k \in \mathbb{N}^+$ *denota el máximo coste.*
- $\mathcal{C}$ *es el conjunto de restricciones como funciones de coste (es decir,* $C_i : \prod_{j \in var(C_i)} D_j \rightarrow \{0, 1, ..., k\}$*) .*

Definición 4.2.2. $S(k) = (\{0, 1, ..., k\}, \oplus, >)$ *es una estructura de valuación, donde:*

- $\{0, 1, ..., k\}$ *es el conjunto de costes, que son números naturales cuyo límite superior es k.*
- $\oplus$ *es el operador suma de costes.*

$$\forall a, b \in \{0, 1, ..., k\},\ a \oplus b = min\{k, a + b\}$$

- $>$ *es el orden estándar entre los números naturales.*

En el capítulo 2 se explicó el concepto de la proyección de una tupla, el cual conviene recordar en esta sección. Dado el subconjunto $\mathcal{X}_t \subseteq \mathcal{X}$, que está compuesto por las variables que componen la tupla t, la proyección de t sobre B (siendo, B un subconjunto de $\mathcal{X}_t$), se denota como $t \downarrow_B$ [17].

Una restricción $C_i \in \mathcal{C}$ está definida para un subconjunto de variables $var(C_i)$, siendo $|var(C_i)|$ la aridad de C_i. Cuando C_i asigna un coste k a una

tupla t (compuesta por $var(C_i)$), significa que t es una tupla no válida para C_i. En caso contrario (el coste asignado es menor que k) t es una tupla válida para C_i y tiene ese correspondiente coste asociado.

El coste total de una tupla t, denotado como $\mathcal{V}(t)$, es la suma de todos los costes aplicables:

$$\mathcal{V}(t) = \bigoplus_{C_i \in \mathcal{C}, var(C_i) \subseteq X_t} C_i(t \downarrow_{var(C_i)})$$

La tupla t es *consistente* si $\mathcal{V}(t) < k$, siendo el principal objetivo encontrar una asignación completa con el mínimo coste.

Ejemplo. En la Figura 4.2 se observa un WCSP $P = \langle\{x, y\}, \{\{v_1, v_2\}, \{v_1, v_2\}\}, S(5), \{C_1, C_2\}\rangle$. Obsérvese que el conjunto de costes es [0,...,5]. Asumamos que $var(C_1) = \{x, y\}$ y $var(C_2) = \{x, y\}$. Los costeas asignados por las restricciones están representados como aristas etiquetadas que conectan los valores de las tuplas que forman parte de la correspondiente restricción. La asignación de costes de C_1 está representada en la Figura 4.2 (a) y la asignación de costes de C_2 está representada en la Figura 4.2 (b).

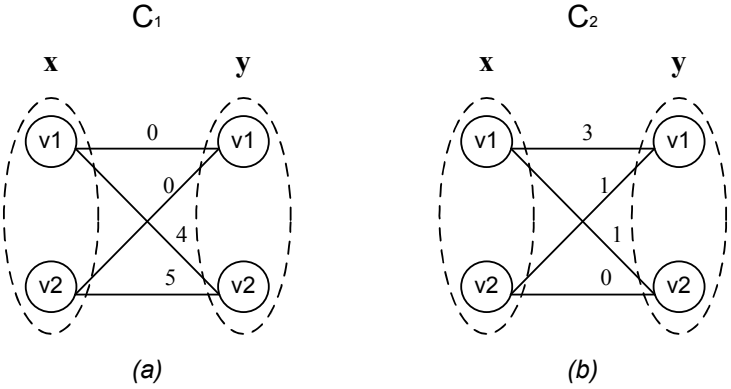

Figura 4.2: WCSP P.

La Tabla 4.1 muestra el conjunto de tuplas de P con sus correspondientes costes asignados por las restricciones y sus valores de $\mathcal{V}(t)$. Además, muestra que tuplas son soluciones de P. Las tuplas $(x = v_1, y = v_2)$ y $(x = v_2, y = v_2)$ no son soluciones de P, ya que sus valores de $\mathcal{V}(t)$ no son inferiores a 5. En cambio, las tuplas $(x = v_1, y = v_1)$ y $(x = v_2, y = v_1)$ son soluciones de P. La mejor solución para P es $(x = v_2, y = v_1)$ porque $\mathcal{V}(x = v_2, y = v_1)$ representa el mínimo coste total.

Tabla 4.1: Conjunto de tuplas de P y sus correspondientes costes.

x	y	Coste C_1	Coste C_2	$\mathcal{V}(t)$	Solución?
v_1	v_1	0	3	3	Sí
v_1	v_2	4	1	5	No
v_2	v_1	0	1	1	Sí
v_2	v_2	5	0	5	No

4.3. Un Ejemplo Juguete de Problema de Scheduling Dinámico

En el ámbito de los problemas de scheduling, pueden surgir diferentes fuentes de incertidumbre: la duración de las actividades puede ser diferente de la planeada inicialmente, los recursos pueden tener menor capacidad de lo esperado (por ejemplo fallos en la maquinaria), dejando inutilizable el schedule construido con la duración inicial de las actividades o la disponibilidad de recursos. Por lo tanto, el schedule debe responder a las situaciones inesperadas y al entorno variable. En este contexto, la generación de schedules robustos desempeña un papel crucial.

A continuación, presentamos un ejemplo juguete de scheduling modelado como un CSP [4]. En este ejemplo, dos actividades A y B tienen que ser finalizadas en 8 horas como máximo: $A_{fin} \leq 8$; $B_{fin} \leq 8$. Además, la actividad B tiene que empezar al menos 3 horas después de que la actividad A finalice: $B_{inicio} - A_{fin} \geq 3$. La duración de la actividad A es de al menos 2 horas: $A_{fin} - A_{inicio} \geq 2$. La duración de la actividad B es de al menos 1 hora: $B_{fin} - B_{inicio} \geq 1$. Este problema de scheduling puede ser modelado como un problema de satisfacción de restricciones $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, donde:

- $\mathcal{X} = \{A_{inicio}, A_{fin}, B_{inicio}, B_{fin}\}$
- $\mathcal{D} = \{D_{A_{inicio}} : [0,8], D_{A_{fin}} : [0,8], D_{B_{inicio}} : [0,8], D_{B_{fin}} : [0,8]\}$
- $\mathcal{C} = \{C_1 : A_{fin} \leq 8$
  $C_2 : B_{fin} \leq 8$
  $C_3 : B_{inicio} - A_{fin} \geq 3$
  $C_4 : A_{fin} - A_{inicio} \geq 2$
  $C_5 : B_{fin} - B_{inicio} \geq 1\}$

Una solución del CSP es: $S_1 = \{A_{inicio} = 0, A_{fin} = 2, B_{inicio} = 5, B_{fin} = 6\}$. La Figura 4.3 muestra el schedule que representa la solución.

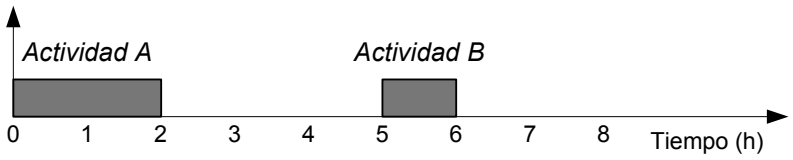

Figura 4.3: Schedule obtenido para el problema.

Como hemos comentado anteriormente, en la vida real, pueden darse cambios en la duración de las actividades respecto a la planeada inicialmente. Por ejemplo, si consideramos un retraso de una hora en la primera actividad ($A_{end} - A_{begin} = 3$). Entonces, la solución inicial obtenida S_1 deja de ser válida, debido a que no satisface la restricción $C_3 : B_{inicio} - A_{fin} \geq 3$.

Volviendo al modelo inicial del problema de scheduling, y teniendo en cuenta que todas las actividades podrían retrasarse, podemos restringir el problema, restringiendo las restricciones asociadas a la duración de una actividad (C_4 y C_5). Por ejemplo, realizaremos un incremento del término constante de cada restricción en una unidad: $C_4 : A_{fin} - A_{inicio} \geq 3$; $C_5 : B_{fin} - B_{inicio} \geq 2$. La solución obtenida para el CSP restringido es $S_2 = \{A_{inicio} = 0, A_{fin} = 3, B_{inicio} = 6, B_{fin} = 7\}$. La Figura 4.4 muestra el schedule que representa la solución. La nueva solución obtenida S_2 es más robusta que la solución original S_1, debido a que si se produce un retraso ($\leq$1 hora) en cualquier actividad, la solución sigue siendo válida.

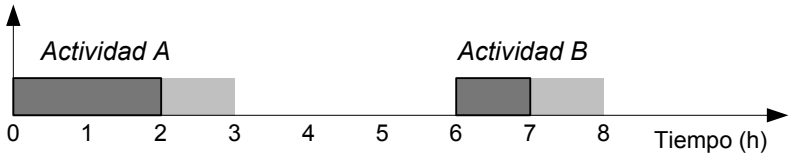

Figura 4.4: Schedule obtenido para el problema restringido.

En problemas de scheduling, la manera más común de generar schedules robustos es mediante la inclusión de tiempos de seguridad (*buffers*) entre actividades. Estos tiempos de seguridad pueden absorber los retrasos y evitan su propagación por el schedule [3]. Sin embargo, también producen un descenso de la optimalidad del schedule, ya que se incrementa el tiempo total en que todas las actividades completan su ejecución (*makespan*).

En la Figura 4.3 se observa el primer schedule, en el cual son necesarias 6 horas, desde el comienzo de la primera actividad, para que las dos actividades (A y B) hayan finalizado. En cambio, en la Figura 4.4 se observa el segundo schedule, en el cual se necesitan 8 horas para que ambas actividades finalicen.

Esto se debe a que el segundo schedule está compuesto de dos buffers de una hora cada uno. Por ello, el segundo schedule es más robusto que el primero. Sin embargo, el primer schedule es más óptimo que el segundo.

En muchos problemas dinámicos, en los cuales existen además criterios de optimalidad, es necesario encontrar una solución de compromiso (*trade-off*) que sea lo más robusta y óptima posible.

4.4. Búsqueda de Soluciones Robustas en DynCSPs

En este apartado, pretendemos explicar los pasos principales del proceso de modelización de los DynCSPs como CSPs ponderados (tanto para *DynCSPs informados* como *DynCSPs no informados*). En posteriores secciones (4.6 y 4.7), será explicado, en detalle, el proceso de modelización para cada tipo de DynCSP.

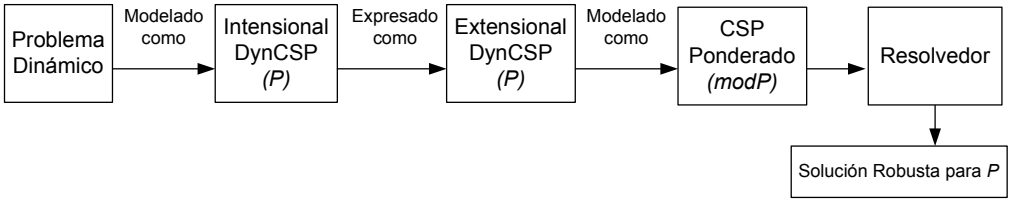

Figura 4.5: Modelado de un DynCSP como un CSP Ponderado.

En la Figura 4.5 se muestran los pasos necesarios para modelar y resolver un DynCSP como un WCSP [4]. Inicialmente, el problema dinámico P es modelado como un CSP intensional. A continuación, se traduce a su representación extensional equivalente. Posteriormente, se genera un CSP ponderado mediante la asignación de los pesos apropiados para las tuplas de cada restricción, obteniendo *modP*. Para ello, utilizaremos el formato de fichero WCSP (ver sección 4.5), que nos permitirá que cualquier resolvedor de CSPs ponderados sea capaz de leer, interpretar y resolver *modP*. El conjunto de soluciones obtenidas para *modP* son las mismas soluciones que para P. El conjunto de soluciones que se obtendrán estarán ordenadas por su robustez. De manera que la mejor solución obtenida para *modP* es la solución considerada más robusta para P.

4.5. Especificación del Formato de Fichero WCSP

El formato WCSP es un formato simple y fácil de analizar sintácticamente por los resolvedores de WCSP. Un fichero en formato WCSP está compuesto de una lista de términos numéricos, a excepción del primero que define el nombre del problema, separados por un espacio, tabulador o fin de línea (ver Tabla 4.2 (derecha)). En lugar de utilizar nombres para hacer referencia a las variables, se utilizan los índices para referenciar a las variables. Lo mismo sucede con los valores del dominio. Todos los índices comienzan en cero. Todas las restricciones se definen en su forma extensional, con su propia lista de tuplas. Se define un valor de coste por defecto para cada restricción, con el objetivo de reducir el tamaño de la lista de tuplas. De esta forma, sólo se especifican las tuplas cuyo coste sea distinto al coste por defecto de la restricción asociada. Todos los costes tienen que ser valores positivos. La estructura del formato es: primero, el nombre del problema y las dimensiones, a continuación la definición de las variables y, por último, la definición de las restricciones. Normalmente, los archivos suelen tener extensión *.wcsp* [29].

Un archivo con formato WCSP comienza con el prólogo:
<Nombre del Problema> $<N>$ $<K>$ $<C>$ $<UB>$, donde

- $<N>$ es el número de variables (entero).
- $<K>$ es el máximo tamaño dominio (entero).
- $<C>$ es el número total de restricciones (entero).
- $<UB>$ es el límite superior de coste total del problema (entero largo).

A continuación del prólogo le siguen las especificaciones de las variables:
<Tamaño del dominio de la variable con índice 0> ... <Tamaño del dominio de la variable con índice N-1>

Las restricciones se especifican como sigue (en una línea):
<Aridad de la restricción>
<Índice de la variable que aparece en el primer campo de la restricción>
...
<Índice de la variable que aparece en el último campo de la restricción>
<Valor del coste por defecto>
<Número de tuplas con un coste distinto al coste por defecto>

Además, para cada tupla (también en una línea cada una):
<Índice del valor asignado a la variable del primer campo de la restricción>

...
<*Índice del valor asignado a la variable del último campo de la restricción*>
<*Coste de la tupla*>

Pueden darse casos en los que hayan varias restricciones con los mismos campos (el resolvedor debería combinarlos en una restricción). La aridad de una restricción puede ser igual a cero. En este caso, no hay tuplas y el valor de coste por defecto se añade al coste total de la solución del problema. Esto puede resultar especialmente útil para representar un límite inferior del coste global del problema. El objetivo es encontrar una asignación de todas las variables con el mínimo coste posible, el cual debe ser estrictamente menor que el coste global *UB*. Las tuplas con un coste mayor o igual que *UB* están prohibidas.

4.6. Soluciones Robustas en DynCSPs Informados

El objetivo de la técnica desarrollada en esta sección, consiste en encontrar soluciones robustas para *DynCSPs informados* [4]. Para lograr este propósito, modelamos el DynCSP (P) como un WCSP ($modP$). La mejor solución para $modP$ es la solución considerada más robusta para P. El nuevo CSP modificado ($modP$) es creado mediante la composición de las restricciones originales de P más la incorporación de un conjunto de nuevas restricciones más restringidas. Las restricciones originales son consideradas restricciones duras, mientras que las nuevas restricciones añadidas son consideradas restricciones blandas. De esta forma, el objetivo es encontrar una solución que satisfaga todas las restricciones duras y el mayor número posible de restricciones blandas.

Las restricciones son representadas como funciones de coste que asignan costes a las tuplas que forman parte de cada restricción. El algoritmo 1 muestra el proceso de modelado de *DynCSPs informados* como WCSPs.

Como se ha comentado anteriormente (capitulo 3), los *DynCSPs informados* están formados por dos funciones de dinamismo: $p(C_i)$ y $d(C_i)$. La función que mide la magnitud de cambio ($d(C_i)$), será utilizada para la generación de nuevas restricciones. La función que mide la probabilidad de cambio ($p(C_i)$) será utilizada en la asignación de los costes de las tuplas no válidas de las nuevas restricciones generadas.

4.6.1. Generación de Nuevas Restricciones

El algoritmo 1 (líneas 6-7) nos muestra como para cada restricción original C_i, se generan un conjunto de restricciones ordenadas por su grado de restringibilidad $\{C_{i1}, C_{i2}, .., C_{iw}\}$, siendo w el número de restricciones que se añaden por cada restricción C_i perteneciente a P. Tanto P como w son proporcionados por el usuario. Cada conjunto de nuevas restricciones generadas, representa posibles modificaciones más restrictivas de una restricción original, en base sus funciones de dinamismo ([7], [32]).

Cada nueva restricción generada C_{ij} está compuesta de un porcentaje de tuplas válidas de la restricción original C_i. De esta manera, C_{ij} es una versión más restrictiva de C_i. Cada restricción original C_i se considera como una restricción dura, mientras que el conjunto de nuevas restricciones generadas $\{C_{i1}, C_{i2}, .., C_{iw}\}$ se consideran restricciones blandas.

A. Propiedades de las Nuevas Restricciones

1. $\forall i \in \{1, .., n\}$ *el conjunto de tuplas válidas de* C_{i1} *está incluido en el conjunto de tuplas válidas de* C_i.

2. $\forall i \in \{1, .., n\}$ *si* C_{i1} *consistente* $\Rightarrow$ C_i *consistente.*

3. $\forall i \in \{1, .., n\}$ *el conjunto de tuplas válidas de* $\{C_{ij} : j \in \{2, .., w\}\}$ *está incluido en el conjunto de tuplas válidas de* $C_{i(j-1)}$.

4. $\forall i \in \{1, .., n\}$ *si* $\{C_{ij} : j \in \{2, .., w\}\}$ *consistente* $\Rightarrow$ $C_{i(j-1)}$ *consistente.*

El número de tuplas que dejan de ser válidas para cada restricción C_{ij} con respecto al número de tuplas válidas de $C_{i(j-1)}$ es: $(d(C_i) *$ *número de tuplas válidas de* $C_i)/w$.

4.6.2. Asignación de Costes

Existen dos tipos de funciones de costes, dependiendo de si la restricción es dura C_i o blanda C_{ij}. El algoritmo 1 nos muestra el coste de las funciones asociadas a las restricciones duras (líneas 3-5) y a las restricciones blandas (líneas 8-11).

El principal objetivo de la función de coste aplicada a una restricción original C_i es el de prohibir las tuplas que no satisfagan una restricción original. De este modo, $C_i(t \downarrow_{var(C_i)})$ asigna un coste k a la tupla t si no

satisface C_i. En el caso contrario, si t satisface C_i, se le asigna un coste 0 a la tupla t. El valor de k es una cota superior proporcionada por el usuario.

La función de coste aplicada a las nuevas restricciones $C_{ij}(t \downarrow_{var(C_{ij})})$ nos permite priorizar entre cada conjunto de nuevas restricciones generadas basándonos en $p(C_i)$. Todas las restricciones de cada conjunto de restricciones ordenadas por su restringibilidad ($\{C_{i1}, C_{i2}, .., C_{iw}\}$), tendrán la misma función de coste. La función de coste $C_{ij}(t \downarrow_{var(C_{ij})})$ asigna un coste 0 a la tupla t si satisface la restricción C_{ij}. Sin embargo, si la tupla t no satisface C_{ij}, le asigna un coste de $p(C_i) * 100$, el cual es siempre más pequeño que k, permitiendo de esta forma a t formar parte de una solución del problema.

La mejor solución para un CSP ponderado es la solución con el coste asociado mas pequeño $\mathcal{V}(t)$. Es por ello que a las tuplas no válidas de las restricciones C_{ij} cuyo $p(C_i)$ asociado es elevado se les asigna un coste elevado. De esta forma, estamos penalizando en mayor medida a estas tuplas, ya que es muy probable que la restricción C_i sufra cambios.

4.6.3. Objetivo

Dado un *DynCSP informado* $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, se lleva a cabo la generación artificial de nuevas restricciones y la asignación de costes a las tuplas de todas las restricciones, con el objetivo de generar un CSP ponderado (*modP*), el cual se puede resolver con cualquier resolvedor de CSPs ponderados. El algoritmo 1 (línea 12) nos muestra la generación de $modP = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}' \rangle$ el cual está compuesto de las variables y dominios originales, y un conjunto de restricciones $\mathcal{C}'$, siendo $\mathcal{C}'$ la unión de las restricciones originales y las nuevas restricciones generadas ($\mathcal{C}' = \bigcup_{i=1}^{n} \{C_i \cup \{C_{i1}, C_{i2}, .., C_{iw}\}\}$).

Generamos *modP* con el objetivo de maximizar el número de restricciones satisfechas de *modP*, basándonos en la prioridad fijada para las nuevas restricciones generadas. Una solución que satisfaga un mayor número de restricciones de *modP* tiene una mayor probabilidad de permanecer siendo válida después de que se produzcan modificaciones en las restricciones del problema original. La prioridad de cada conjunto de nuevas restricciones generadas se expresa en términos del coste asignado a las tuplas que no satisfacen esas nuevas restricciones. Si este coste es elevado, significa que la restricción tiene una prioridad alta, ya que la mejor solución para un CSP ponderado es la tupla t que tenga el mínimo $\mathcal{V}(t)$ asociado.

Todas las soluciones obtenidas para *modP* son las mismas soluciones del problema original P. El valor de $\mathcal{V}(t)$ representa el nivel de robustez de la solución para el problema original P. El algoritmo 1 (líneas 14-16) muestra la solución (si existe) de *modP*. Esta solución es la solución más robusta para

el *DynCSP informado* original, de acuerdo con sus funciones de dinamismo.

Algoritmo 1: Modelado de un DynCSP Informado como WCSP

Data: A CSP(P), $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, funciones de dinamismo $p(C_i)$ y $d(C_i)$, parámetro w y k.
Result: Solución Robusta Sol y su robustez $\mathcal{V}(Sol)$.
begin

```
1   foreach C_i ∈ C do
2       foreach t ∈ C_i do
3           if t satisface C_i then
4               C_i(t ↓_var(C_i)) = 0;
            else
5               C_i(t ↓_var(C_i)) =k;
6       foreach  j ∈ {1, .., w} do
7           Generar {C_ij} basándonos en d(C_i);
8           foreach t ∈ C_i do
9               if t satisface C_ij then
10                  C_ij(t ↓_var(C_ij)) = 0;
                else
11                  C_ij(t ↓_var(C_ij)) = p(C_i) * 100;
12  Generar modP = ⟨X, D, C'⟩ donde, C' = {C_i ∪ {C_i1, C_i2, .., C_iw}} ;
13  Sol ← Resolver modP;
14  if ∃ Sol then
15      return (Sol, V(Sol));
    else
16      return (No existe solución);
```

end

4.6.4. Ejemplo

A continuación veremos un ejemplo de un *DynCSP informado* y de su proceso de modelización como un CSP ponderado.

Sea P un *DynCSP informado* con dos variables x_0 y x_1, con dominios $D_0 : \{3..7\}$ y $D_1 : \{2..6\}$ respectivamente. Las restricciones y sus correspondientes funciones de dinamismo son:

- $C_1(0.2, 0.2) : x_0 + x_1 \leq 12$

- $C_2(0.8, 0.4) : x_1 + x_0 \geq 6$
- $C_3(0.4, 0.2) : x_1 - x_0 \leq 2$
- $C_4(0.2, 0.4) : x_0 - x_1 \leq 4$

Cada restricción es etiquetada con dos números reales. El primer número entre los paréntesis representa la probabilidad de dinamismo de la restricción ($p(C_i)$) y el segundo la magnitud del cambio de la restricción ($d(C_i)$).

En la Figura 4.6 [5] podemos ver la representación de P. En este ejemplo concreto, P está compuesto por restricciones lineales.

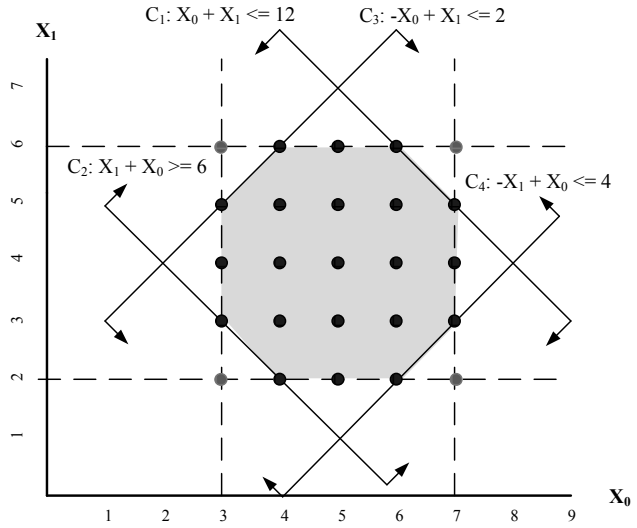

Figura 4.6: Ejemplo de CSP [5].

Siguiendo los pasos representados en la Figura 4.5, el primer paso es convertir el DynCSP intensional a su representación extensional. Si nosotros asumimos que $w = 1$, entonces el WCSP resultante ($modP$) estará compuesto de 8 restricciones (ver Tabla 4.2). El primer paso del modelado consiste en la creación de un conjunto de restricciones más restrictivas $\{C_{i1}, C_{i2}, .., C_{iw}\}$, basándose en las funciones de dinamismo.

En la Tabla 4.2 (derecha) puede verse representado $modP$ en el formato de fichero WCSP. La primera restricción del problema (C_1) está compuesta por 2 variables, siendo x_0 la primera variable y x_1 la última. Esta restricción tiene 24 tuplas válidas con un coste asociado de 0. El resto de tuplas no satisfacen la restricción C_1 por lo que tienen como coste asociado k. La primera tupla válida asociada a la restricción C_1 es $(x_0 = 4, x_1 = 2)$ y la última tupla es $(x_0 = 7, x_1 = 6)$. Además, el coste asociado a todas estas tuplas es 0. La

última restricción (C_{41}) también está compuesta por las mismas 2 variables. Sin embargo, esta restricción tiene 14 tuplas válidas con coste asociado de 0. Mientras que las tuplas que no satisfacen la restricción C_{41} tienen un coste por defecto de 20. La primera y la última tuplas válidas son: $(x_0 = 3, x_1 = 2)$ y $(x_0 = 7, x_1 = 5)$ respectivamente.

Una solución del problema original P obtenida por un resolvedor genérico de CSPs, podría ser $(x_0 = 3, x_1 = 3)$. Tras modelar el problema como un CSP ponderado (obteniendo $modP$) y resolverlo, averiguamos que dicha solución tiene un coste de 100 y tan sólo satisface 6 de las 8 restricciones de $modP$. Sin embargo, la solución que proporciona el resolvedor de CSPs ponderados es $(x_0 = 6, x_1 = 5)$, la cual tiene un coste de 0 y satisface todo el conjunto de restricciones, tanto las originales como las nuevas restricciones más restrictivas generadas (en total 8 restricciones).

Tal y como hemos comentado anteriormente, el resolvedor de WCSPs encuentra como solución la tupla t con el mínimo valor de $\mathcal{V}(t)$ asociado. Cuanto más grande es el valor de $\mathcal{V}(t)$ asociado a una solución t, menos robusta es la solución. Asimismo, el número de restricciones más restringidas $\{C_{i1}, C_{i2}, .., C_{iw}\}$ que una solución satisface es otra unidad de medida de la robustez. En este ejemplo, la solución encontrada para $modP$ satisface todas las restricciones y eso significa que es una de las soluciones más robustas para P.

4.7. Soluciones Robustas en DynCSPs No Informados

En esta sección se explicará la búsqueda de soluciones robustas para *DynCSPs no informados*. Este tipo de DynCSP no tiene información asociada acerca del dinamismo de sus restricciones debido a que el usuario no conoce información sobre la probabilidad y la magnitud de cambio de las restricciones. Como no conocemos esta información, hemos considerado que la solución con la mayor distancia global a todas las restricciones es la solución más robusta, ya que tal solución tiene una alta probabilidad de permanecer válida después de que se produzcan cambios más restrictivos en las restricciones del problema.

En este contexto, nos centraremos en DynCSPs con restricciones aritméticas de la forma: $Ax\{<, \leq, >, \geq\}l$, donde x representa las variables que forman parte de la restricción, A los coeficientes de las variables y l el término

Tabla 4.2: DynCSP original (P) (izquierda) y CSP ponderado ($modP$) (derecha).

Variables: $X_0..X_1$ Dominios: $D_0 : 3-7$ $D_1 : 2-6$	ejemplo.wcsp 2 2 8 8 100000000 8 7
$C_1(0.8, 0.4) : x_1 + x_0 \geq 6$	2 1 0 100000000 24 2 4 0 ... 6 7 0
C_{11} : 60 % de tuplas de C_2	2 1 0 80 14 3 4 0 ... 6 6 0
$C_2(0.2, 0.2) : x_0 + x_1 \leq 12$	2 0 1 100000000 24 3 2 0 ... 7 5 0
C_{21} : 80 % de tuplas de C_1	2 0 1 20 19 3 2 0 ... 7 5 0
$C_3(0.4, 0.2) : x_1 - x_0 \leq 2$	2 1 0 100000000 19 2 3 0 ... 6 7 0
C_{31} : 80 % de tuplas de C_3	2 1 0 40 15 2 4 0 ... 6 7 0
$C_4(0.2, 0.4) : x_0 - x_1 \leq 4$	2 0 1 100000000 24 3 2 0 ... 7 6 0
C_{41} : 60 % de tuplas de C_4	2 0 1 20 14 3 2 0 ... 7 5 0

independiente.

La técnica propuesta consiste en el modelado de un DynCSP (P) como CSP ponderado, generando *modP*. La mejor solución obtenida para *modP* será considerada la solución más robusta para P. El CSP ponderado es creado con las restricciones de P representadas como funciones de coste, las cuales asignan costes a las tuplas que están asociadas a cada restricción. El algoritmo 2 nos muestra el modelado de *DynCSPs no informados* como WCSPs.

4.7.1. Asignación de Costes

En el algoritmo 2 (líneas 3-5) se observan las funciones de coste asociadas a cada restricción C_i. La principal utilidad de las funciones de coste es la de asignar distintas prioridades a las tuplas que componen el conjunto de tuplas válidas para cada restricción, basándonos en la distancia de las tuplas a la restricción dada. Si la distancia de la tupla t a la restricción C_i es alta, significa que t tiene una probabilidad elevada de continuar permaneciendo válida después de que se produzcan modificaciones en C_i.

Gráficamente, la distancia de la tupla a la restricción puede ser vista como la proyección de t en la dimensión n, siendo n el número de variables que componen la restricción. Por ejemplo, si nosotros estamos tratando con restricciones binarias, la proyección será en dos dimensiones, y podremos medir la distancia desde el punto representado por la tupla a la línea que representa la restricción.

Numéricamente, podemos obtener esta distancia reemplazando el valor de las variables para una tupla en la restricción. El número obtenido después de calcular la ecuación representa esta distancia. Si la distancia es 0, la tupla se encuentra gráficamente situada en la línea que representa la restricción. Esta tupla es una de las menos robustas para esta restricción, ya que ante cualquier modificación más restrictiva que se produzca en la restricción, la tupla pasará a ser no válida. Es por este motivo que el coste asignado a este tipo de tuplas es el coste máximo. El caso opuesto, en el que que la tupla tiene la máxima distancia posible denotada por $MaxD_i$, el coste asignado a esta tupla es 0, ya que como se ha mencionado anteriormente, la mejor solución para un CSP ponderado es la solución cuyo valor de $\mathcal{V}(t)$ sea el mínimo posible.

El coste máximo asociado a cada restricción se calcula como el cuadrado de la máxima distancia posible desde cualquier tupla permitida a la restricción. Calculamos el cuadrado de la distancia debido a que es una manera de penalizar las tuplas que se encuentran situadas próximas a alguna restricción. La asignación de coste a una tupla válida t se hace de forma gradual entre

el 0 y el máximo coste calculado: $C_i(t \downarrow_{var(C_i)}) = (Distancia\ t - MaxD_i)^2$. Sin embargo, $C_i(t \downarrow_{var(C_i)})$ asigna un coste de k a la tupla t si no satisface la restricción C_i.

4.7.2. Objetivo

Después de haber generado el CSP ponderado *modP*, mediante la asignación de costes a todas las tuplas de todas las restricciones, lo resolvemos mediante un resolvedor de WCSPs. El algoritmo 2 nos muestra la creación (línea 6) y resolución (línea 7) de *modP*. El objetivo del algoritmo es encontrar una solución cuya distancia global a todas las restricciones del problema sea la más alta posible. La prioridad asignada a las tuplas válidas se basa en la distancia de las tuplas a cada restricción, y se expresa en términos del coste asignado a las tuplas para cada restricción. El coste asignado es más pequeño a medida que la distancia es mayor.

Todas las soluciones obtenidas para *modP* son soluciones de *P*. Además, el orden del valor de $\mathcal{V}(t)$ representa el orden de la robustez de las soluciones para el CSP original, siendo la solución con el mínimo valor de $\mathcal{V}(t)$, la solución más robusta para el CSP original (P). El algoritmo 2 (líneas 8-10) muestra la solución (si existe) de *modP*.

Algoritmo 2: Modelado de un DynCSP No Informado como WCSP

Data: Un CSP(P), $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, funciones $p(C_i)$ y $d(C_i)$ y parámetro w.
Result: Solución robusta Sol y su robustez $\mathcal{V}(Sol)$
begin

```
1   foreach C_i ∈ C do
2       foreach t ∈ C_i do
3           if t es válida para C_i then
4               C_i(t ↓_var(C_i)) = (DistanciaA − M)^2;
            else
5               C_i(t ↓_var(C_i)) =k;
6   Generar modP = ⟨X, D, C⟩ ;
7   Sol ← Resolver modP;
8   if ∃ Sol then
9       return (Sol, V(Sol));
    else
10      return (No existe solución);
```

end

4.7.3. Ejemplo

A continuación se presenta el ejemplo anterior, pero sin información asociada a cada restricción, por lo que se trata de un *DynCSP no informado*. Así, no conocemos información de como cambiarán las restricciones.

En la Figura 4.7 podemos ver la representación gráfica del problema. Existen 21 soluciones para el problema. Si nosotros no sabemos nada acerca del dinamismo de las restricciones, ¿Qué solución es la solución más robusta? En este ejemplo, se puede ver que la solución más robusta podría ser la solución situada en el centro del espacio de soluciones ($x_0 = 5, x_1 = 4$) (ver Figura 4.7).

La idea principal es asignar costes a las tuplas permitidas para cada restricción. Los costes son calculados por medio de la ecuación $C_i(t \downarrow_{var(C_i)}) = (Distancia\ t\ - MaxD_i)^2$.

Tabla 4.3: Ejemplo de la restricción C_2 para *modP* (izquierda) y las soluciones de *modP* con sus costes asociados (derecha).

$C_1 : x_0 + x_1 \leq 12$
2 0 1 100000000 24
3 2 0
3 3 1
3 4 4
3 5 9
3 6 16
4 2 1
4 3 4
4 4 9
4 5 16
4 6 25
5 2 4
5 3 9
5 4 16
5 5 25
5 6 36
6 2 9
6 3 16
6 4 25
6 5 36
6 6 49
7 2 16
7 3 25
7 4 36
7 5 49

Solución	Coste
x0=3 x1=3	84
x0=3 x1=4	80
x0=3 x1=5	84
x0=4 x1=2	84
x0=4 x1=3	72
x0=4 x1=4	68
x0=4 x1=5	72
x0=4 x1=6	84
x0=5 x1=2	80
x0=5 x1=3	68
x0=5 x1=4	64
x0=5 x1=5	68
x0=5 x1=6	80
x0=6 x1=2	84
x0=6 x1=3	72
x0=6 x1=4	68
x0=6 x1=5	72
x0=6 x1=6	84
x0=7 x1=3	84
x0=7 x1=4	80
x0=7 x1=5	84

En este ejemplo, la distancia máxima de cualquier tupla a cualquier restricción es de 7 unidades. Por lo tanto, el máximo coste que es posible asignar a las tuplas válidas es de 49. El mínimo coste es de 0. En la Tabla 4.3 (izquierda), podemos ver la representación de la segunda restricción C_2 para el *modP*. La primera línea contiene los parámetros de la restricción y el resto de líneas son las tuplas permitidas para esta restricción. Para las líneas que representan las tuplas permitidas, el primer número de la línea es el valor de la variable x_0, el segundo número es el valor de la variable x_1 y el tercero es el coste asignado a la tupla. Por ejemplo, el coste de la tupla $(x_0 = 4, x_1 = 4)$ para la primera restricción es calculado como $((12-4-4)-7)^2$, cuyo resultado es 9 (ver Tabla 4.3). El mismo proceso se desarrolla para todas las restricciones del CSP. Finalmente, utilizamos un resolvedor de CSPs ponderados, obteniendo todas las soluciones y sus costes globales.

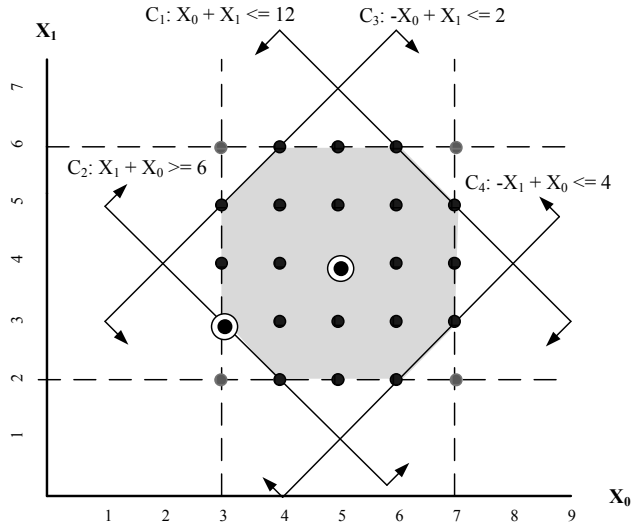

Figura 4.7: Ejemplo de CSP con dos de sus soluciones [5].

Como se puede observar en la Tabla 4.3 (derecha) y Figura 4.7 [5], el resolvedor de CSPs ponderados encuentra la solución t con el mínimo $\mathcal{V}(t)$. Esta solución corresponde a la solución situada en el centro del poliedro compuesto por las restricciones del problema: $(x_0 = 5, x_1 = 4)$, siendo su valor de $\mathcal{V}(t)$ de 64.

Sin embargo, la solución encontrada por un resolvedor usual de CSPs sería $(x_0 = 3, x_1 = 3)$, que tiene un valor de $\mathcal{V}(t)$ de 84 (ver Tabla 4.3 (derecha)), que es el máximo coste global. Es decir, es una de las soluciones menos robustas del problema. Podemos observar en la Figura 4.7 como la solución $(x_0 = 3, x_1 = 3)$ está gráficamente situada en la restricción de

$C_2 : x_1 + x_0 \geq 6$. Por ello, es muy probable que esta solución deje de ser válida cuando sucedan cambios más restrictivos en la restricción C_2.

Capítulo 5

Evaluación

5.1. Introducción

La evaluación de las técnicas explicadas en este trabajo ha consistido en el análisis de la robustez de las soluciones obtenidas en los CSPs que hemos modificado, así como su comparación con la robustez de las soluciones que se obtienen en los CSPs originales.

La evaluación se ha realizado para *DynCSPs informados* y *DynCSPs no informados*, en problemas dinámicos generados aleatoriamente y benchmarks existentes en la literatura.

En este capítulo también evaluaremos el efecto de los parámetros del DynCSP y de las funciones de dinamismo en la robustez de las soluciones obtenidas. Comparándolo además con soluciones obtenidas en los CSPs originales.

A lo largo de este capítulo introduciremos distintas unidades de medida de la robustez para *DynCSPs informados* y *DynCSPs no informados*, las cuales son medidas alternativas al coste total asociado a las soluciones del WCSP modelado.

5.2. DynCSPs Informados

Para la evaluación de *DynCSPs informados* hemos utilizado una medida de la robustez alternativa al coste total asociado a las soluciones ($\mathcal{V}(t)$, siendo t una solución del DynCSP). La medida alternativa para los *DynCSPs informados* se basa en la cantidad de nuevas restricciones añadidas al WCSP que la solución satisface. Cuanto más alto sea este número, más robusta es la

solución, ya que tiene mayor probabilidad de continuar siendo válida aunque se produzcan cambios en las restricciones del problema original.

El tiempo máximo de resolución de cada CSP ha sido fijado a 600 segundos.

5.2.1. Problemas Aleatorios

La evaluación se ha desarrollado mediante la generación de 100 instancias aleatorias para cada problema, siendo el parámetro $w = 1$. Los problemas han sido generados aleatoriamente incrementando el número de variables, la talla de los dominios, el número de restricciones (h_C) y la restringibilidad de las restricciones (r_C).

En la Tabla 5.1 podemos observar los porcentajes del número de restricciones añadidas satisfechas para las soluciones obtenidas para el CSP original (*CSP Original*) y para el WCSP modelado (*WCSP*).

Tabla 5.1: Porcentaje de nuevas restricciones añadidas que satisfacen las soluciones encontradas para un resolvedor de CSPs y para el WCSP propuesto.

				Restricciones añadidas satisfechas	
Variables	Dominio	h_C	r_C	*WCSP*	*CSP Original*
15	60	30	0.4	70 %	46.6 %
30	60	30	0.4	93.3 %	56 %
60	60	30	0.4	96 %	50 %
30	15	60	0.9	96.6 %	65 %
30	30	60	0.9	95 %	65 %
30	60	60	0.9	95 %	65 %
60	30	15	0.4	93.3 %	46.6 %
60	30	30	0.4	93.3 %	50 %
60	30	60	0.4	93.3 %	46.6 %
60	60	60	0.9	95 %	58.3 %
60	60	60	0.7	90 %	53.3 %
60	60	60	0.4	90 %	50 %

El porcentaje obtenido representa una medida de la robustez de la solución. Cuanto más alto es este porcentaje, mas robusta es una solución, ya que tiene mayor probabilidad de continuar siendo válida aunque se produzcan cambios en las restricciones del problema.

Se puede observar en la Tabla 5.1 que el porcentaje de nuevas restricciones más restrictivas que satisfacen las soluciones encontradas para el WCSP modelado, es mayor a medida que aumenta el número de variables del problema.

Esto se debe a que existen más tuplas válidas ya que el número de variables es más alto. Sin embargo, no puede observarse ninguna otra relación significativa con los otros parámetros del problema (talla de los dominios, número de restricciones y restringibilidad de las restricciones).

La conclusión más importante que se deduce de este análisis es que para todos los casos analizados, las soluciones encontradas después de realizar el modelado de WCSP son mucho más robustas que las soluciones encontradas para el CSP original, ya que como se observa en la Tabla 5.1, las soluciones encontradas tras el modelado WCSP tienen los porcentajes de nuevas restricciones añadidas satisfechas mucho más altos que los de las soluciones encontradas para el CSP original.

5.2.2. Benchmarks

En la literatura no existen benchmarks de *DynCSPs informados*. Por lo tanto, hemos considerado benchmarks de WCSPs y les hemos añadido funciones de dinamismo a las restricciones de los problemas. Las instancias utilizadas provienen de dos benchmarks de WCSPs: *Academics* y *Planning* [16]. El formato de los benchmarks es el formato de archivo WCSP.

Para esta evaluación, se han realizado dos análisis de la robustez diferentes, basados en: el número de adiciones de restricciones que se realizan y en las funciones de dinamismo. En ambos análisis hemos omitido los problemas que sólo tienen una solución en el problema original, ya que carecen de interés para nuestro estudio. Además, hemos eliminado las restricciones originales blandas de los problemas debido a que nuestra técnica no ha sido desarrollada para problemas de satisfacción y optimización de restricciones (Constraint Satisfaction and Optimization Problem - CSOP).

A. Análisis de Robustez Basado en el Número de Nuevas Restricciones Creadas

En el primer análisis desarrollado, nuestro objetivo ha sido determinar la eficacia de nuestra técnica y analizar la influencia del parámetro w (número de restricciones que se añaden por cada C_i del CSP original) en la robustez de la solución obtenida. Para cada problema se han generado diez *DynCSPs informados* distintos mediante la creación de funciones de dinamismo aleatorias.

En la Tabla 5.2 se pueden observar los problemas que han sido analizados, el número de restricciones duras (h_C) que lo componen y el número

de nuevas restricciones modificadas que se han añadido (m_C). Este número está directamente relacionado con el parámetro w. Sin embargo, hay un punto de saturación para el número de nuevas restricciones modificadas que es posible crear para cada restricción. Esto se debe a que no permitimos la creación de una nueva restricción modificada que tenga las mismas tuplas validas que otra restricción modificada añadida. Para cada problema, se muestra el número de restricciones modificadas satisfechas (s_C) para las soluciones obtenidas para el CSP original (Ori) y para el WCSP modelado ($WCSP$), así como la diferencia (D) entre ambos valores.

La Figura 5.1 muestra la diferencia de nuevas restricciones satisfechas (D) para las soluciones obtenidas para el CSP original (Ori) y para el WCSP modelado ($WCSP$), utilizando diferentes valores de w. Cuatro de los problemas más representativos de la Tabla 5.2 han sido representados en la gráfica.

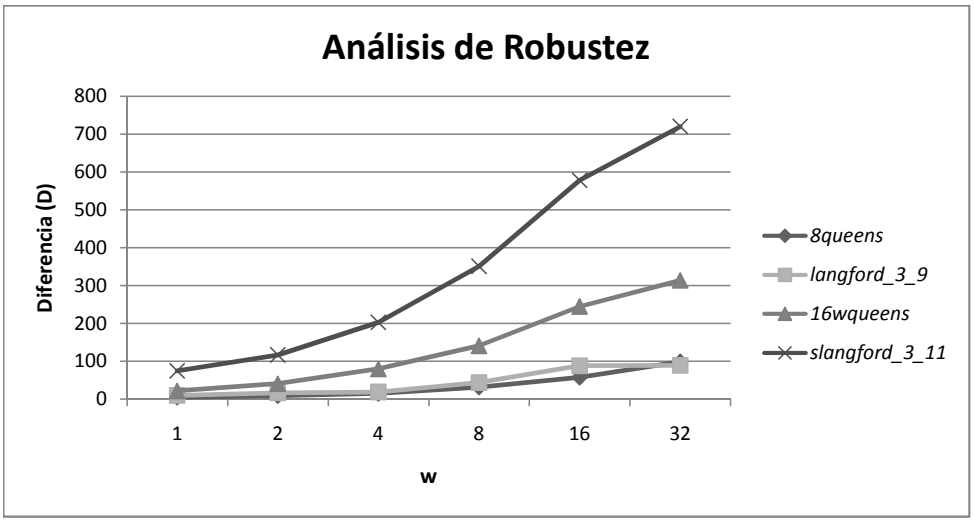

Figura 5.1: Análisis de Robustez basado en w.

Los resultados obtenidos para el análisis de la robustez muestran que para todos los problemas, nuestro método obtiene soluciones más robustas que las soluciones obtenidas mediante la resolución del CSP original. Esto se puede determinar debido a que las soluciones del WCSP modelado satisfacen un mayor número de nuevas restricciones modificadas que las soluciones del CSP original. La diferencia entre ambas cifras (D) se incrementa en problemas en los que el número de restricciones y el número de tuplas válidas para cada restricción son elevados. Por ejemplo, para el problema *slangford_3_11* se obtiene la mayor D, ya que las soluciones obtenidas con nuestra técnica satisfacen un total de 720 nuevas restricciones modificadas más que las soluciones

Tabla 5.2: Análisis de Robustez basado en w.

		$w=1$			$w=2$			$w=4$		
Problema	Sol	m_C	s_C	D	m_C	s_C	D	m_C	s_C	D
4queens	*Ori*	6	2	1	9	2	1	14	4	0
(h_C=4)	*WCSP*		3			3			4	
8queens	*Ori*	28	14	5	56	20	9	112	34	15
(h_C=28)	*WCSP*		19			29			49	
langford_2_4	*Ori*	29	12	2	54	17	2	87	21	3
(h_C=32)	*WCSP*		14			19			24	
langford_3_9	*Ori*	369	176	10	737	275	17	1441	453	19
(h_C=369)	*WCSP*		186			292			472	
16wqueens	*Ori*	120	61	22	240	87	41	476	142	80
(h_C=120)	*WCSP*		83			128			222	
slangford_3_11	*Ori*	550	270	75	1100	408	117	2180	680	203
(h_C=550)	*WCSP*		345			525			883	
driverlog01cc	*Ori*	39	14	10	66	16	11	93	17	11
(h_C=472)	*WCSP*		24			27			28	
logistics01bc	*Ori*	140	41	8	214	52	9	272	52	10
(h_C=2222)	*WCSP*		49			61			62	
mprime01ac	*Ori*	141	59	12	255	79	20	433	110	29
(h_C=12264)	*WCSP*		71			99			139	
rovers02ac	*Ori*	63	21	2	98	21	3	130	25	4
(h_C=5029)	*WCSP*		23			24			29	
		$w=8$			$w=16$			$w=32$		
Problema	Sol	m_C	s_C	D	m_C	s_C	D	m_C	s_C	D
4queens	*Ori*	17	4	1	21	4	1	25	4	2
(h_C=4)	*WCSP*		5			5			6	
8queens	*Ori*	218	56	32	392	94	58	590	102	99
(h_C=28)	*WCSP*		32			152			201	
langford_2_4	*Ori*	103	23	6	106	26	1	97	31	2
(h_C=32)	*WCSP*		29			27			33	
langford_3_9	*Ori*	2643	693	44	4197	957	88	4755	981	89
(h_C=369)	*WCSP*		737			1045			1070	
16wqueens	*Ori*	903	245	141	1555	351	245	2067	432	314
(h_C=120)	*WCSP*		386			596			746	
slangford_3_11	*Ori*	4106	1112	351	6960	1656	578	9167	1999	720
(h_C=550)	*WCSP*		1463			2234			2719	
driverlog01cc	*Ori*	93	15	14	92	15	12	87	16	13
(h_C=472)	*WCSP*		29			27			29	
logistics01bc	*Ori*	283	52	11	272	48	15	280	51	11
(h_C=2222)	*WCSP*		63			63			62	
mprime01ac	*Ori*	646	138	40	792	155	60	818	153	63
(h_C=12264)	*WCSP*		178			215			216	
rovers02ac	*Ori*	138	21	4	138	23	3	134	25	6
(h_C=5029)	*WCSP*		25			26			31	

obtenidas para el CSP original.

En la Tabla 5.2 y Figura 5.1, también podemos observar que w está directamente relacionada con D. Sin embargo, cuando w toma el valor del punto de saturación, no es posible crear nuevas restricciones modificadas (m_C), por lo que D deja de aumentar. Cada problema tiene un punto de saturación diferente, dependiendo del número de tuplas válidas para las restricciones del problema. Por ejemplo, el problema *langford_ 3_ 9* tiene un punto de saturación de $w = 16$, ya que D deja de incrementar para $w > 16$ (véase Figura 5.1). Sin embargo, la Figura 5.1 muestra que los otros tres problemas no tienen un punto de saturación menor o igual que 32, ya que para $w = 32$, la D ha incrementado en relación a valores más pequeños de w.

B. Análisis de Robustez Basado en las Funciones de Dinamismo

En la segunda evaluación realizada, nuestro objetivo ha sido analizar el efecto de las funciones de dinamismo en la robustez de las soluciones obtenidas. Con este fin, hemos escogido dos problemas del conjunto de benchmarks: *driverlog01cc* y *slangford_ 3_ 11*, siendo el parámetro $w = 1$. Para cada caso se han generado diez *DynCSPs informados* diferentes, mediante la elección aleatoria de diferentes tuplas válidas para las nuevas restricciones modificadas.

En la Tabla 5.3 se puede observar el número de restricciones modificadas generadas (m_C) para los dos problemas analizados. También se muestra el número de restricciones modificadas satisfechas (s_C) para las soluciones obtenidas para el CSP original (Ori) y para el WCSP modelado ($WCSP$), así como la diferencia entre estos valores (D). Las funciones de dinamismo $p(C_i)$ y $d(C_i)$ se representan como p y d.

La Figura 5.2 muestra la diferencia de nuevas restricciones satisfechas (D) para las soluciones obtenidas para el CSP original (Ori) y para el WCSP modelado ($WCSP$), siendo $p(C_i) = 0.2$.

Se puede observar en la Tabla 5.3 y la Figura 5.2 que la diferencia de nuevas restricciones satisfechas (D) aumenta cuando aumentamos $d(C_i)$. Cuando esto sucede, el porcentaje de tuplas que pasan a ser no válidas para una nueva restricción modificada, aumenta también. Sin embargo, hay un punto de saturación para ambos problemas, que es $d(C_i) = 0.8$. En este punto, el número de nuevas restricciones satisfechas (s_C) disminuye debido a que el grado de restringibilidad de las nuevas restricciones generadas para el WCSP modelado es tan alta que es poco probable que las soluciones satisfagan un número elevado de tales restricciones.

Tabla 5.3: Análisis de Robustez basado en $p(C_i)$ y $d(C_i)$.

		driverlog01cc	p=0.2		p=0.4		p=0.6		p=0.8	
d	Sol	m_C	s_C	D	s_C	D	s_C	D	s_C	D
0.2	*Ori*	26	16	9	18	6	17	7	18	7
	WCSP		25		24		24		25	
0.4	*Ori*	43	19	11	20	11	22	8	19	12
	WCSP		30		31		30		31	
0.6	*Ori*	49	11	13	14	11	14	10	13	11
	WCSP		24		25		24		24	
0.8	*Ori*	49	3	8	4	7	3	8	3	8
	WCSP		11		11		11		11	
		slangford_3_11	p=0.2		p=0.4		p=0.6		p=0.8	
d	Sol	m_C	s_C	D	s_C	D	s_C	D	s_C	D
0.2	*Ori*	550	437	76	437	75	440	72	435	78
	WCSP		513		512		512		513	
0.4	*Ori*	550	330	91	331	93	331	89	333	91
	WCSP		421		427		420		424	
0.6	*Ori*	550	215	94	217	95	219	91	224	86
	WCSP		309		312		310		310	
0.8	*Ori*	550	107	75	108	73	112	65	178	71
	WCSP		182		181		117		107	

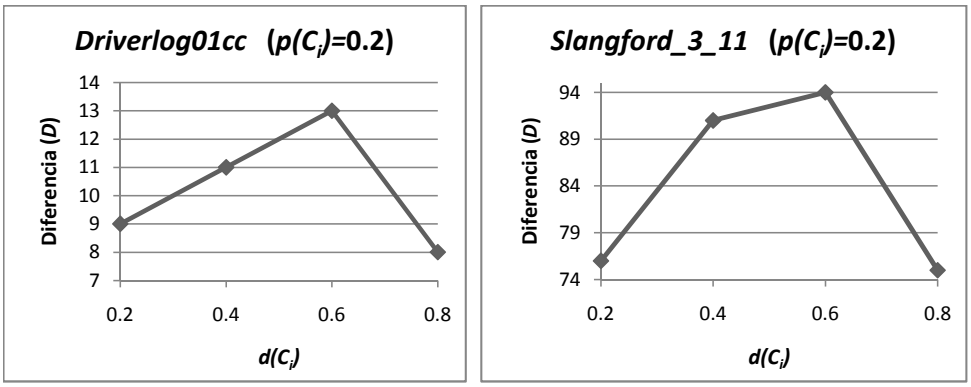

Figura 5.2: Análisis de Robustez basado en $d(C_i)$ para $p(C_i) = 0.2$.

Para ambos problemas el nivel más elevado de D se alcanza para $d(C_i) = 0,6$. Además, observando la Tabla 5.3, podemos afirmar que $p(C_i)$ no tiene una relación directa con D. La función $p(C_i)$ se utiliza en la asignación de los costes de las tuplas no válidas para las restricciones C_{ij}. Por lo tanto, la función $p(C_i)$ tiene efecto en el coste de las soluciones obtenidas ($\mathcal{V}(t)$) pero no en el número de restricciones modificadas satisfechas (s_C).

5.3. DynCSPs No Informados

En el caso de *DynCSPs no informados* no se posee ninguna información acerca del dinamismo de las restricciones. Es por este motivo que la robustez de las soluciones no puede medirse en base a las probabilidades y magnitudes de cambio de las restricciones del problema ($p(C_i)$ y $d(C_i)$), ya que no conocemos tal información, por lo que no podemos utilizar la misma unidad de medida de la robustez (número de nuevas restricciones modificadas satisfechas) que hemos utilizado para los *DynCSPs informados*.

Además de la unidad de medida de la robustez que consiste en la medida del coste total de las soluciones ($\mathcal{V}(t)$), utilizaremos una medida alternativa de la robustez para *DynCSPs no informados*. Esta medida consiste en sumar el número de tuplas válidas cuyos valores forman parte de la solución ($\sum_{i=1}^{n}$ *Num. Tuples*), siendo n el número de variables.

La idea principal de esta medida de la robustez alternativa, es que si sabemos que hay una gran cantidad de tuplas cuyas variables tienen asignados los mismos valores de la solución, esto significa que la solución es robusta. La idea se basa en probabilidades, debido a si hay un cambio en alguna restricción, es más probable que el valor de una variable que tenga más ocurrencias en las tuplas válidas del DynCSP, siga siendo válido como solución del problema. Por lo tanto, cuanto mayor es esta unidad de medida, más robusta será la solución.

La evaluación se ha desarrollado mediante la generación de 100 instancias aleatorias para cada problema, siendo el parámetro $w = 1$. Los problemas han sido generados aleatoriamente incrementando el número de variables, la talla de los dominios, el número de restricciones (h_C) y la restringibilidad de las restricciones (r_C).

En la Tabla 5.4 podemos observar los resultados de la robustez de las soluciones, utilizando las dos unidades de medida ($\sum_{i=1}^{n}$ *Num. Tuples* y $\mathcal{V}(t)$) para las soluciones obtenidas para el WCSP modelado ($WCSP$) y para el CSP original (Ori).

Se puede observar en la Tabla 5.4, que los resultados para las soluciones obtenidas tras el modelado WCSP son mejores que los de las soluciones obtenidas para el CSP original. Para todos los casos, el número de tuplas válidas cuyos valores forman parte de la solución, son más elevados para las soluciones encontradas tras el modelado WCSP. Esto significa que las soluciones encontradas con nuestro método son más robustas que las soluciones encontradas para el CSP original.

Además, es interesante analizar cómo los parámetros del problema influyen en las unidades de medida de la robustez. La suma del número de tuplas válidas cuyos valores forman parte de la solución, es mayor cuando el número

Tabla 5.4: Media de la suma del número de tuplas y media de la suma de los costes asociados a las tuplas para las soluciones obtenidas mediante el modelado WCSP y de los CSPs originales.

				$\sum_{i=1}^{n}$ Num. Tupples		$\mathcal{V}(t)$	
Variables	Dominio	*h_C*	*r_C*	*WCSP*	*Ori*	*WCSP*	*Ori*
15	60	30	0.4	141	138	51315	111819
30	60	30	0.4	73	68	26930	117574
60	60	30	0.4	36	33	15429	135253
30	15	60	0.9	43	39	47	225
30	30	60	0.9	84	75	202	1559
30	60	60	0.9	168	151	815	14456
60	30	15	0.4	9	8	336	10201
60	30	30	0.4	19	17	983	9644
60	30	60	0.4	38	34	2110	9604
60	60	60	0.9	82	72	573	18386
60	60	60	0.7	77	69	12347	101034
60	60	60	0.4	70	66	32372	126414

de tuplas válidas cuyos valores forman parte de la solución para cada restricción es mayor, es decir, cuando los tamaños de dominios son más grandes o la restringibilidad de las restricciones es menor. En estos casos, esta medida se incrementa en mayor grado para las soluciones robustas del problema.

La medida del coste total de las soluciones ($\mathcal{V}(t)$) no depende de la cantidad de tuplas válidas para cada restricción, pero sí de la distancia de la tupla a la restricción. Es por esta razón que si los tamaños de dominios son más grandes o la restringibilidad de las restricciones es menor, el valor de $\mathcal{V}(t)$ también es más elevado.

Además, ambas unidades de medida de la robustez tienen una dependencia directa con el número de restricciones del problema y una dependencia indirecta con el número de variables del problema.

Capítulo 6

Conclusiones y Futuros Trabajos

6.1. Aportaciones

En este trabajo, hemos presentado dos métodos de búsqueda de soluciones robustas, basados en el modelado de los *DynCSPs* como WCSPs. Además, hemos introducido dos tipos distintos de DynCSPs: *DynCSPs informados* y *DynCSPs no informados*. Los *DynCSPs informados* son DynCSPs con información asociada acerca del dinamismo de las restricciones. En cambio, los *DynCSPs no informados* no tienen ningún tipo de información asociada acerca del dinamismo de las restricciones.

Además, en este trabajo, se ha realizado un análisis de los conceptos de estabilidad y robustez de las soluciones de DynCSPs, haciendo hincapié en sus diferencias. De esta manera, hemos definido la robustez de una solución como la medida de la persistencia de la solución ante modificaciones en el problema original. Así, una solución de un DynCSP es robusta si tiene una alta probabilidad de permanecer válida ante futuros cambios en el problema.

Con el fin de resolver *DynCSPs informados*, hemos introducido una técnica que consiste en el modelado del *DynCSP informado* como un WCSP que está compuesto por las restricciones del DynCSP original y por un conjunto de nuevas restricciones generadas. Las nuevas restricciones generadas son modificaciones más restrictivas de las restricciones del DynCSP original. Además, se realiza la asignación de costes para las tuplas de todas las restricciones del WCSP, teniendo en cuenta las funciones de dinamismo asociadas al *DynCSP informado*.

La técnica introducida para resolver *DynCSPs no informados* también consiste en el modelado del *DynCSP no informado* como WCSP, sin embargo el WCSP sólo está compuesto por las restricciones del DynCSP original. La

asignación de costes para las tuplas se realiza en relación a la "distancia" de cada tupla a cada restricción.

Tras la evaluación realizada tanto en los *DynCSPs informados* como en los *DynCSPs no informados*, podemos afirmar que para todos los casos analizados (problemas aleatorios y benchmarks de la literatura), las soluciones obtenidas con nuestros métodos son igual o más robustas que las soluciones obtenidas del CSP original.

En comparación con las técnicas mencionadas en el estado del arte, las técnicas desarrolladas en este trabajo presentan como principal ventaja sobre las técnicas correctivas que evitan la pérdida de las soluciones calculadas siempre que sea posible. La principal ventaja que incorporan nuestras técnicas sobre la técnica proactiva es que nuestras técnicas no incrementan el coste computacional.

Resumiendo, los beneficios que aportan tanto los *DynCSPs informados* y *DynCSPs no informados*, como los métodos de búsqueda de soluciones robustas para éstos son:

- *Fiable.* Se puede garantizar que se obtienen soluciones robustas para los DynCSPs mediante las técnicas de modelado como WCSP que hemos desarrollado.

- *Práctico.* Las propuestas realizadas utilizan algoritmos eficientes para resolver el WCSP modelado. De esta manera, no incrementan el coste computacional de resolución de los DynCSPs. Además evitan la pérdida de las soluciones calculadas siempre que sea posible.

En particular, los *DynCSPs informados* y su correspondiente método de busca de soluciones robustas, también tienen como beneficio:

- *Específico.* Los *DynCSPs informados* permiten concretar en la modelización de problemas dinámicos en los cuales se conoce información acerca de los futuros posibles cambios en las restricciones del problema. De esta forma, se orienta la búsqueda de soluciones robustas en base a dicha información, obteniendo mejores resultados.

En particular, los *DynCSPs no informados* y su correspondiente método de busca de soluciones robustas, también tienen como beneficio:

- *Genérico.* Los *DynCSPs no informados* permiten la modelización de problemas dinámicos en los cuales no se conoce información acerca de los futuros cambios en las restricciones. De esta manera, se orienta la búsqueda de soluciones robustas hacia cualquier posible cambio restrictivo de las restricciones del problema original.

En la literatura no se encuentran unidades de medida de la robustez generales. Por este motivo, además de proporcionar las unidades de medida de la robustez especificas para nuestra técnica (coste total de las soluciones), hemos intentado utilizar otras unidades alternativas más genéricas: número de nuevas restricciones modificadas satisfechas (para *DynCSPs informados*) y suma del número de tuplas válidas que tienen los mismos valores asignados a las variables que los de la solución (para *DynCSPs no informados*).

De la evaluación realizada para *DynCSPs informados*, hemos extraído como conclusiones principales:

- Cuanto mayor es el número de variables, mayor es la cantidad de restricciones modificadas satisfechas por las soluciones encontradas en nuestro modelo en comparación con las soluciones encontradas en el CSP original.

- Cuanto mayor es el número de nuevas restricciones modificadas que generamos por cada restricción del problema original, la diferencia entre la robustez de las soluciones obtenidas para WCSP modelado con respecto a las soluciones obtenidas para el CSP original aumenta. Sin embargo, existe un punto de saturación en el que este incremento deja de producirse.

- Cuanto mayor es $d(C_i)$ para las restricciones, mayor es la diferencia de nuevas restricciones satisfechas entre las soluciones del WCSP modelado y el CSP original. También existe un punto de saturación, en el que el aumento de tal diferencia deja de producirse y se convierte en un descenso.

De la evaluación realizada para *DynCSPs informados*, hemos extraído como conclusiones principales:

- Cuanto mayores son los tamaños de los dominios y/o menores son los grados de restringibilidad de las restricciones, la diferencia entre la robustez de las soluciones obtenidas para WCSP modelado con respecto a las soluciones obtenidas para el CSP original aumenta.

Las técnicas y modelos desarrollados en este trabajo pueden ser de gran utilidad en muchos problemas dinámicos reales. Para este tipo de problemas, el objetivo de su resolución va más allá de la eficiencia de los algoritmos empleados y el usuario desea obtener diferentes niveles de robustez de las soluciones. Además, nuestros métodos presentan una ventaja sobre las estrategias correctivas y otras estrategias proactivas debido a que no aumentan el coste computacional de la resolución del DynCSP.

6.2. Líneas Abiertas

Existen diversas líneas de investigación abiertas hacia los DynCSPs en la que todavía queda mucho camino y trabajo por realizar. La relevancia de los DynCSPs viene dada porque se utilizan para modelar numerosos problemas de la vida real. Sin embargo, se está muy lejos todavía del objetivo de encontrar modelos y técnicas que sean capaces de resolver de forma eficaz y eficiente cualquier problema dinámico sometido a cambios en los parámetros del problema.

A lo largo del tiempo, se han propuesto varias técnicas para resolver los DynCSPs. La mayoría de estas técnicas son estrategias correctivas. Sin embargo, no existen apenas en la literatura, trabajos orientados a técnicas proactivas, área en la cual se enmarca este trabajo. Por lo tanto, son muchos los desafíos y retos abiertos entorno a esta área.

Las principales cuestiones pendientes para este trabajo, son:

1. *Aplicación en escenarios reales.* Un aspecto muy importante es el poder identificar y modelar diversos problemas de entornos reales, como *DynCSPs informados* y *DynCSPs no informados* dependiendo de si se conoce información o no, referente al dinamismo de las restricciones del problema. Algunos problemas que pueden considerarse apropiados para este estudio debido a las propiedades de dinamismo de sus restricciones son:

   - Problemas de asignación de horarios: Por ejemplo, en ámbitos ferroviarios se deben asignar tiempos de salidas y llegadas para las distintas estaciones y trenes de una red ferroviaria, considerando posibles retrasos e incidencias en los trenes, lo que produciría cambios en las restricciones asociadas [3].
   - Problemas de decisión: Por ejemplo, en ámbitos de supermercados en los que es necesario averiguar la cantidad de producto que se solicita en un pedido, teniendo en cuenta factores como la caducidad y demanda del producto, siendo este último, un factor dinámico que afecta y produce cambios en las restricciones asociadas.

2. *Estudio de DynCSPs en los que se producen otro tipo de cambios.* Considerar más cambios que pueden darse además de los cambios en las restricciones: pérdida de determinados valores del dominio para determinadas variables del DynCSP, aparición de nuevas variables, eliminación de variables, etc.

3. *Desarrollo de heurísticas.* Implementación de heurísticas para algoritmos de búsqueda de soluciones robustas para DynCSPs. Adaptar los

algoritmos de búsqueda mediante heurísticas que sean capaces de orientar la búsqueda hacia valores de variables que tengan más probabilidad de ser los más robustos, teniendo en cuenta que estas heurísticas deben ser eficaces y eficientes, de manera que no incrementen el coste computacional.

4. *Desarrollar técnicas multi-objetivo.* En muchos problemas de la vida real existen varios criterios de optimización que se pretenden conseguir. Por este motivo, se han desarrollado muchas técnicas multi-objetivo que buscan soluciones que maximicen diversos criterios. Dado el elevado número de casos de la vida real dinámicos que persiguen algún criterio de optimización, consideramos que un aspecto muy importante es el desarrollo de técnicas multi-objetivo que combinen la búsqueda de soluciones robustas con otros criterios de optimización.

6.3. Publicaciones Asociadas

- L. Climent, M. A. Salido , F. Barber

  Finding Robust Solutions by Weighted CSPs

  International Journal of Innovative Computing, Information and Control (IJICIC) (enviado).

- L. Climent, M. A. Salido , F. Barber

  Finding Robust Solutions in Constraint Satisfaction Problems

  The 15th International Conference on Principles and Practice of Constraint Programming (CP 2009), Doctoral Consortium. pp:25-30, 2009.

- L. Climent, M. A. Salido , F. Barber

  Robust Solutions in Changing Constraint Satisfaction Problems

  The 23rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE). Part I, pp. 752-761

Springer LNAI 6096, ISBN: 978-3-642-13021-2 ISSN: 0302-9743, 2010.

- L. Climent, F. Barber, M. A. Salido, L. Ingolotti
  Robustness and capacity in scheduling: Application to railway timetabling
  IBERAMIA 2008, Workshop on Planning, Scheduling and Constraint Satisfaction. pp: 87-96. ISBN: 972886207-5, 2008.

- M. A. Salido, L. Climent, F. Barber
  A Tool for Finding Stable Solutions in Constraint Satisfaction Problems
  CAEPIA 2009, Workshop on Planning, Scheduling and Constraint Satisfaction, 2009.

- L. Climent, M. A. Salido , F. Barber
  Stable Solutions in Constraint Satisfaction Problems
  The 13th Spanish Conference for Artificial Intelligence (CAEPIA), Doctoral Consortium. 2009.

- L. Climent, M. A. Salido , F. Barber
  Stability of Solutions in Constraint Satisfaction Problems
  The 12th International Conference of the Catalan Association for Artificial Intelligence (CCIA). pp.301-309. ISBN: 978-1-60750-061-2 ISSN: 0922-6389, 2009.)

Bibliografía

[1] A. Algergawy, E. Schallehn, and G. Saake. Fuzzy constraint-based schema matching formulation.

[2] C. Bessiere. Arc-consistency in dynamic constraint satisfaction problems. In *Proceedings AAAI'91*, 1991.

[3] L. Climent, F. Barber, M. Salido, and L. Ingolotti. Robustness and capacity in scheduling: Application to railway timetabling. In *Workshop on Planning, Scheduling and Constraint Satisfaction (IBERAMIA)*, pages 87–96.

[4] L. Climent, M. Salido, and F. Barber. Finding robust solutions by weighted csps. *International Journal of Innovative Computing, Information and Control (IJICIC) (enviado).*

[5] L. Climent, M. Salido, and F. Barber. Finding Stable Solutions in Constraint Satisfaction Problems. In *The 15th International Conference on Principles and Practice of Constraint Programming Doctoral Program Proceedings*, page 25, 2009.

[6] L. Climent, M. Salido, and F. Barber. Stability of Solutions in Constraint Satisfaction Problems. In *Proceeding of the 2009 conference on Artificial Intelligence Research and Development: Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence*, pages 301–309. IOS Press, 2009.

[7] L. Climent, M. Salido, and F. Barber. Stable Solutions in Constraint Satisfaction Problems. In *The 13th Spanish Conference for Artificial Intelligence (CAEPIA), Doctoral Consortium.*, 2009.

[8] L. Climent, M. Salido, and F. Barber. Robust Solutions in Changing Constraint Satisfaction Problems. In *The 23rd International Conference*

on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pages 752–761, Part 1. Springer, 2010.

[9] R. Dechter and A. Dechter. Belief maintenance in dynamic constraint networks. *In Proc. of the 7th National Conference on Artificial Intelligence (AAAI-88)*, pages 37–42, 1988.

[10] R. Dechter and I. Meiri. Experimental evaluation of preprocessing algorithms for constraints satisfaction problems. *Artificial Intelligence*, 68:211–241, 1994.

[11] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint network. *Artificial Intelligence*, 49:61–95, 1991.

[12] H. El Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic sheduling. In *Constraints*, pages 359–388. Springer Netherlands, 2000.

[13] E. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29:24–32, 1982.

[14] D. Frost and R. Dechter. Look-ahead value orderings for constraint satisfaction problems. *In Proc. of IJCAI-95*, pages 572–578, 1995.

[15] P. Geelen. Dual viewpoint heuristic for binary constraint satisfaction problems. *In proceeding of European Conference of Artificial Intelligence (ECAI'92)*, pages 31–35, 1992.

[16] S. Givry. Soft csp benchmarks. `http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP`.

[17] S. D. Givry, J. Larrosa, P. Meseguer, and T. Schiex. Solving max-sat as weighted csp. pages 363–376. Springer Verlag, 2003.

[18] R. Haralick and E. G. Increasing tree efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–314, 1980.

[19] E. Hebrand, B. Hnich, and T. Walsh. Super CSPs. Technical report, 2003.

[20] E. Hebrard, B. Hnich, and T. Walsh. Super solutions in constraint programming. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. CPAIOR-04*, pages 157–172, 2004.

[21] E. Jen. Stable or robust? what's the difference? *Complexity*, 8:12–18, 2003.

[22] N. Keng and D. Yun. A planning/scheduling methodology for the constrained resources problem. *In Proceeding of IJCAI-89*, pages 999–1003, 1989.

[23] J. Larrosa, P. Meseguer, and T. Schiex. Maintaining reversible DAC for Max-CSP. *Artificial Intelligence*, 107(1):149–163, 1999.

[24] J. Larrosa and T. Schiex. Solving weighted csp by maintaining arc consistency. *Artificial Intelligence*, 159:1–26, 2004.

[25] X. Luo, J. Ho-man Lee, H. Leung, and N. Jennings. Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation. *Fuzzy sets and systems*, 136(2):151–188, 2003.

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

[27] S. Minton, M. Johnston, A. Philips, and P. Laird. A heuristic repair method for constraint-satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[28] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[29] L. Otten. Wcsp file format specification. `http://graphmod.ics.uci.edu/group/WCSP_file_format`.

[30] P. Purdom. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence*, 21:117–133, 1983.

[31] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*. Elsevier Science Ltd, 2006.

[32] M. Salido, L. Climent, and F. Barber. A Tool for Finding Stable Solutions in Constraint Satisfaction Problems. In *Workshop on Planning, Scheduling and Constraint Satisfaction (CAEPIA)*, 2009.

[33] T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.

[34] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *In Proceedings of the 14th IJCAI*, pages 631–637, 1995.

[35] G. Verfaillie and T. Schiex. Solution reuse in dynamic constraint satisfaction problems. In *In Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 307–312, 1994.

[36] R. Wallace and E. Freuder. Stable solutions for dynamic constraint satisfaction problems. In *Proc. 4th International Conference on Principles and Practice of Constraint Programming*, pages 447–461. Springer, 1998.

[37] R. Wallace, D. Grimes, and E. Freuder. Solving dynamic constraint satisfaction problems by identifying stable features. *International Joint Conferences on Artificial Intelligence*, pages 621–627, 2009.

[38] L. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.