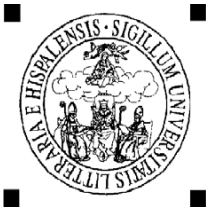
UNIVERSIDAD DE SEVILLA



Departamento de Matemática Aplicada 1

SOBRE ALGUNAS CLASES POLINOMIALES DE SATISFACIBILIDAD

APLICACIONES A LA RESOLUCIÓN DE PROBLEMAS GEOMÉTRICOS

José Inácio de Jesus Rodrigues

Sevilla, 2009

UNIÃO EUROPEIA
Fundo Social Europeu [1]

[1] Acção 5.3 da Medida 5 - Formação Avançada de Docentes do Ensino Superior

UNIVERSIDAD DE SEVILLA

Departamento de Matemática Aplicada 1

Sobre Algunas Clases Polinomiales de Satisfacibilidad

Aplicaciones a la Resolución de Problemas Geométricos

Memoria presentada por
José Inácio de Jesus Rodrigues
para optar al grado de
Doctor en Matemáticas
por la Universidad de Sevilla

V.º B.º del Director:

Fdo.: Dr. D. José Ra. Portillo Fernandez

Sevilla, Marzo del 2009

À Lúcia

Mar Português

Ó mar salgado, quanto do teu sal
São lágrimas de Portugal!
Por te cruzarmos, quantas mães choraram,
Quantos filhos em vão rezaram!
Quantas noivas ficaram por casar
Para que fosses nosso, ó mar!

Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu,
Mas foi nele que espelhou o céu.

Fernando Pessoa, in Mensagem (1934)

Resumen

La Geometría Computacional posee un vasto campo de aplicaciones en las áreas de las Ciencias de Información Geográfica, Electrónica y Computación, entre muchas otras. En estos campos, los problemas de etiquetado de mapas, trazados de circuitos integrados o creación de diagramas de flujos y organigramas son ejemplos particulares de problemas cuyas instancias son en muchos casos reducibles al problema de satisfacibilidad proposicional (SAT). Es decir, su resolución puede ser conseguida a partir de la resolución de determinadas instancias de SAT.

SAT es un problema NP-completo muy estudiado, fundamental en Ciencias de la Computación. A pesar de su intratabilidad, se conocen varias (sub)clases del problema que son resolubles en tiempo polinomial. En el presente trabajo presentamos una nueva clase de satisfacibilidad polinomial: PURL (*PropUnit Removable Literal*). La cual es, en cierto sentido, una superclase respecto a las clases de satisfacibilidad polinomiales identificadas como *bien conocidas* en el trabajo de Franco y van Gelder sobre este tipo de clases [FG03]. A partir de esta nueva clase se define una jerarquía de clases de satisfacibilidad, PURL $\subset$ 2PURL $\subset \cdots \subset$ kPURL, en la línea de los trabajos sobre jerarquías de Gallo y Scutella.

PURL se define a partir del concepto de literal p-eliminable. Mediante este concepto se pueden simplificar las cláusulas de una fórmula proposicional. Dado que la fórmula simplificada y la original son lógicamente equivalentes, el procedimiento de identificación y exclusión de literales eliminables puede ser adoptado como preprocesamiento para cualquier instancia de SAT. Como los literales p-eliminables son reconocibles en tiempo lineal, el preprocesamiento opera en tiempo polinomial.

Con una codificación adecuada mediante variables lógicas de Boole, se reducen las instancias del problema UC-4P de etiquetado de puntos alineados con etiquetas cuadradas a SAT obteniéndose la clase UC-4P-SAT, subclase de PURL. Por otro lado, reduciendo las instancias del problema EOSC de emparejamiento ortogonal simple en el cilindro se obtiene la clase de satisfacibilidad EOSC-SAT, subclase de 2PURL. Además del interés específico de estos problemas, la técnica utilizada puede ser adoptada para la resolución en tiempo polinomial de otros problemas.

Las principales contribuciones del presente trabajo son, entre otras, la definición de una nueva clase de satisfacibilidad polinomial y la aplicación del concepto de literal p-eliminable a la resolución de problemas prácticos. La clase PURL contiene propiamente a todas las clases polinomiales *bien conocidas*. La metodología desarrollada a partir del concepto del literal p-eliminable para la resolución de las instancias de PURL, (y, por tanto, del problema de satisfacibilidad proposicional de las instancias de todas estas clases) constituye un abordaje unificador.

En el presente trabajo también se muestra que el algoritmo de reconocimiento y eliminación de literales p-eliminables puede ser utilizado eficientemente en la resolución de algunas instancias del problema general de satisfacibilidad proposicional.

Para concluir, en la parte final de la memoria se utiliza el concepto de literal p-eliminable para la resolución de dos problemas geométricos irresueltos hasta ahora. La misma metodología puede aplicarse en la resolución de problemas similares, de naturaleza geométrica o no.

Resumo

A Geometria Computacional possui um vasto campo de aplicações nas áreas das Ciências da Informação Geográfica, Electrónica e Computação, entre muitas outras. Nestes domínios, problemas de etiquetado de mapas, traçados de circuitos integrados ou criação de diagramas de fluxos e organogramas constituem exemplos particulares de problemas cujas instâncias são redutíveis ao problema da satisfação proposicional (SAT). Pelo que a sua resolução pode ser conseguida a partir da resolução de determinadas instâncias de SAT.

SAT é um bem conhecido problema NP-completo, fundamental nas Ciências da Computação. Apesar da sua intratabilidade, são conhecidas diversas (sub)classes resolúveis em tempo polinomial. No presente trabalho apresenta-se uma nova classe de satisfação polinomial: PURL (*PropUnit Removable Literal*). A qual, em certo sentido, constitui uma superclasse relativamente às classes de satisfação polinomial identificadas no trabalho de Franco e van Gelder como *bem conhecidas* [FG03]. A partir desta nova classe define-se uma hierarquia de classes de satisfação, PURL $\subset$ 2PURL $\subset \cdots \subset$ kPURL, na linha dos trabalhos sobre hierarquias de Gallo e Scutella.

PURL é definida a partir do conceito de literal p-eliminável, pelo qual as cláusulas de uma fórmula proposicional se podem simplificar. Dado que a fórmula simplificada e a original são logicamente equivalentes, o procedimento de identificação e exclusão de literais p-eliminàveis poderá ser adoptado como metodologia de pré-processamento para qualquer instância de SAT. Como os literais p-eliminàveis são reconhecíveis em tempo linear, este pré-processamento é efectuado em tempo polinomial.

Para uma adequada codificação com recurso a variáveis lógicas booleanas, reduzindo as instâncias do problema de etiquetado de pontos alinhados com etiquetas quadradas a instâncias de SAT, obtém-se a classe UC-4P-SAT, subclasse de PURL. Por outro lado, reduzindo as instâncias do problema de emparelhamento ortogonal simples no cilindro obtém-se a classe de satisfação proposicional EOSC-SAT, subclasse de 2PURL. Para além do interesse específico destes problemas, a técnica utilizada

poderá ser adoptada na resolução em tempo polinomial de outros problemas.

Como principais contributos do presente trabalho salientam-se, entre outros, a definição da nova classe de satisfação polinomial e a aplicação do conceito de literal p-eliminável na resolução de problemas práticos. PURL, contém propriamente cada uma das classes polinomiais *bem conhecidas*. Pelo que a metodologia desenvolvida a partir do conceito de literal p-eliminável (para resolução das instâncias de PURL, e portanto do problema da satisfação proposicional das instâncias de todas estas classes) constitui uma abordagem unificadora.

No presente trabalho, mostra-se ainda que o algoritmo de reconhecimento e exclusão de literais p-eliminàveis pode ser utilizado eficientemente na resolução de algumas instâncias do problema da satisfação proposicional.

A concluir, na parte final do presente trabalho utiliza-se o conceito de literal p-eliminável para a resolução de dois problemas geométricos que estavam por resolver. Esta mesma metodologia poderá ser aplicada na resolução de problemas similares, de natureza geométrica ou não.

Agradecimentos

Por fim, concluída! Em retrospectiva, é inevitável recordar as inúmeras viagens a Sevilha, reuniões, aulas, trabalhos, avanços, recuos, mas também o conforto do inestimável apoio de professores e colegas, família e amigos. Sem a vossa ajuda não teria sido possível! Na impossibilidade material de agradecer a cada um individualmente, seguramente necessitaria de várias páginas e mesmo assim correria o risco de esquecer alguém, a todos o meu profundo agradecimento, obrigado. Contudo, não posso deixar de referir algumas pessoas cujo contributo foi determinante, sem o qual esta dissertação nunca teria sido escrita.

Ao Prof. José Ra. Portillo, meu orientador, o meu profundo agradecimento pela supervisão e direcção desta dissertação, pelas longas horas de paciência lendo e traduzindo textos em português, pela disponibilidade, generosidade e amizade ao longo de todos estes anos.

Aos professores do programa de doutoramento em Matemática Discreta agradeço os ensinamentos ministrados através das várias e interessantes disciplinas durante o período lectivo. Um agradecimento especial aos professores Alberto Márquez, meu tutor, Juan António Mesa, Francisco Ortega e T.B. Boffey pela oportunidade de colaborar num modelo para resolução de problemas de localização utilizando Sistemas de Informação Geográfica.

À Universidade do Algarve, à Escola Superior de Tecnologia e ao PRODEP agradeço os apoios concedidos.

Obrigado Lúcia, pela tua infinita paciência, pelo teu apoio incondicional. Pelo que não fizeste por ti mas que fizeste por mim. Obrigado pelo teu amor.

Obrigado Nuno e Francisco, pela vossa alegria e pelo vosso carinho. Nem sempre foi fácil adormecer sem a *nossa* história e por vezes sem o beijo de boa noite. Esses nossos momentos, os jogos que ficaram por jogar, as brincadeiras por brincar, lamento ter perdido.

Introducción

El problema de SATISFACIBILIDAD PROPOSICIONAL (SAT) es un problema NP-completo. De hecho, es el primer problema NP-completo conocido [coo71], extremadamente relevante por su interés teórico en el campo de la complejidad computacional y por sus múltiples y diversas aplicaciones prácticas.

De la extensa bibliografía sobre aplicaciones de SAT podemos extraer algunos ejemplos de aplicaciones: síntesis lógica [BVM84, GP95], colocación y encaminamiento de circuitos electrónicos [dev89, WR97], test de circuitos [lar92, SBV96], planeamiento [CB94, FG90], telecomunicaciones [WR96], matemática discreta [gel02, SK99], robótica [CD86], procesamiento de texto [HGW93], juegos sudoku [LO06, ari05] y arquitectura de computadores [RB83]. La profundización del conocimiento del problema de la SATISFACIBILIDAD PROPOSICIONAL y el desarrollo e implementación de algoritmos eficientes para su resolución constituyen dos vertientes interdependientes sobre las cuales se desenvuelve, en la actualidad, la mayor parte del trabajo de investigación en SAT.

El problema de la SATISFACIBILIDAD PROPOSICIONAL ha merecido un continuo y creciente interés por parte de la comunidad científica como fácilmente se reconoce por la cantidad y calidad de los eventos sobre el tema. Son excelentes ejemplos de esta afirmación la *International Conference on Theory and Applications of Satisfiability Testing*, con ediciones anuales desde el año 2000, cuya décima edición transcurrió entre el 28 y el 31 de Mayo de 2007 en Lisboa[2] y las distintas competiciones internacionales entre varios tipos de algoritmos para la resolución de instancias de SAT (*International SAT Competitions*[3]).

La comunidad de investigadores en SAT tiene presencia viva a través de Internet, especialmente en las páginas del sitio SATLive[4], con cerca

[2] `http://sat07.ecs.soton.ac.uk`
[3] `http://www.satcompetition.org/`
[4] `http://www.satlive.org/`

de 400 personas registradas y en los sitios SAT-Ex[5], SATLIB[6], SAT4J[7] y JSAT[8], que son de gran utilidad para los interesados en este problema que pretendan desarrollar investigación sobre SAT y sus aplicaciones. SATlive está mantenido por Daniel le Berre y contiene enlaces actualizados a artículos y programas enviados por sus miembros y una sección para el anuncio de conferencias sobre el tema. SAT-Ex se creó con el objetivo de recoger y divulgar información sobre experiencias en la resolución de problemas de satisfacibilidad proposicional. SATLIB cumple un papel de divulgación de *benchmarks* e *solver releases*. SAT4J es una librería de programas y rutinas desarrolladas en el lenguaje Java orientados al problema de satisfacibilidad proposicional. JSAT es una revista *peer reviewed*, donde se encuentran artículos relacionados con *sat*.

A pesar de las capacidades reveladas por algunos algoritmos para la resolución de SAT, consiguiendo la resolución eficiente de instancias con centenas de millar de variables y millones de cláusulas, no se conoce ningún algoritmo polinomial que resuelva SAT. Ni se espera, en general, que tal algoritmo exista. Esta es la convicción de gran parte de la comunidad científica que conjetura que $P \neq NP$. Sin embargo, la relación entre estas dos clases de problemas, P (*easy to find*) y NP (*easy to check*), formalizadas independientemente por Leonid Levin y Stephen Cook en 1971 [coo71], continúa siendo un problema abierto. A título de curiosidad referiremos que el *Clay Mathematics Institute* ofreció en Mayo de 2000 un premio de un millón de dólares americanos por la resolución de cada uno de siete (más) importantes problemas en el campo de la Matemática que han resistido a sucesivos intentos de resolución y que constituyen un desafío para el nuevo milenio: entre estos siete problemas se encuentra dilucidar si P es igual o si está estrictamente incluido en NP. Esto es una muestra de la importancia atribuida al tema de la complejidad computacional, tema al que SAT va inexorablemente asociado.

Independientemente de esta importante cuestión acerca de la complejidad de SAT, existen algunas clases de satisfacibilidad proposicional cuyas instancias son resolubles por algoritmos polinomiales. 2SAT es un ejemplo paradigmático de este tipo de clase, cuyas instancias están formadas por cláusulas con dos literales. En la bibliografía sobre SAT pueden encontrarse otras clases de satisfacibilidad polinomiales, habiendo sido referenciadas las más importantes por Franco y Gelder en su artículo *A perspective on certain polynomial-time solvable clases of satisfiability* [FG03], el cual es una referencia inevitable en el pre-

[5] `www.lri.fr/~simon/satex/satex.php3`

[6] `http://www.satlib.org/`

[7] A satisfiability library for Java, `http://www.sat4j.org/`

[8] Journal on satisfiability, boolean modelling and computations, `http://www.jsat.org/`

sente trabajo, debido a que el resultado principal de éste es presentar una nueva clase de satisfacibilidad polinomial, PURL (*PropUnit Removable Literal*). Esta nueva clase contiene propiamente a todas las clases polinomiales que Franco y Gelder designaron como *bien conocidas (well-known)* [FG03].

En el presente trabajo definimos y usamos el concepto de *literal eliminable* y de algunos de sus casos particulares para resolución de ciertos problemas geométricos. A partir de este concepto se define también una nueva clase de satisfacibilidad polinomial PURL (*PropUnit Removable Literal*). Un literal eliminable en una cláusula puede ser excluido obteniéndose una fórmula proposicional lógicamente equivalente a la inicial. Pero, para una fórmula proposicional 3SAT, determinar si cada uno de los literales de una de sus cláusulas es eliminable equivale a determinar su no satisfacibilidad. Por lo tanto, si 3SAT es un problema intratable también lo es el problema de identificar literales eliminables.

La definición de PURL se basa en el concepto de *literal p-eliminable* que es un caso particular de literal eliminable. Por lo tanto, un literal p-eliminable en una cláusula también puede ser eliminado obteniéndose una fórmula proposicional lógicamente equivalente a la inicial. Pero, al contrario del que ocurre con los literales eliminables en general, la identificación y exclusión de literales p-eliminables se puede hacer en tiempo polinomial por el algoritmo ELIMINALITERALES. Si existe una cláusula con todos sus literales p-eliminables, la fórmula devuelta por este algoritmo contiene la cláusula nula, por lo que que ni la fórmula simplificada ni la original son satisfacibles. Para una instancia SAT arbitraria, ELIMINALITERALES es un algoritmo incompleto, i.e., si la fórmula proposicional simplificada contiene la cláusula nula, sabemos que la fórmula original no es satisfacible pero en el caso contrario, nada se puede deducir.

En el presente trabajo se prueba que existen clases de satisfacibilidad (con un interés práctico para la resolución de determinados problemas) en las que sus instancias son satisfacibles si y solamente si la fórmula retornada por el algoritmo ELIMINALITERALES no contiene a la cláusula nula. Estas fórmulas son subclases de PURL.

Es sabido que, en el ámbito de la SATISFACIBILIDAD PROPOSICIONAL se pretende decidir si una dada fórmula proposicional es satisfacible y, en su caso, encontrar soluciones o probar que dichas soluciones no existen y, por lo tanto, que la dicha fórmula no es satisfacible (que es normalmente la parte más difícil del problema SAT). Los conceptos de literal eliminable y p-eliminable, explorando la existencia de bloqueos a la satisfacibilidad de las instancias, permiten el estudio y la resolución de uno de los aspectos críticos de SAT. Además de otras, la clase PURL contiene las instancias que contienen alguna cláusula con todos sus literales p-eliminables. Por lo tanto, la no satisfacibilidad de estas fórmulas

proposicionales del problema de SATISFACIBILIDAD puede ser determinada por algoritmos polinomiales. Queda por determinar en el presente trabajo un método genérico de reconocimiento de las instancias de esta nueva clase (PURL), de un modo similar a lo que ocurre con otra clase polinomial SLUR (*Single Lookahead Unit Resolution*), la cual probaremos que es una subclase propia de PURL, y para la que tampoco se conoce ningún algoritmo que permita el reconocimiento de sus instancias en tiempo polinomial.

Un intento de desarrollar un método para la resolución de instancias del problema de satisfacibilidad proposicional sin necesidad del reconocimiento previo nos lleva al algoritmo SOLELIMLIT. Éste es un algoritmo polinomial, incompleto, cuyas respuestas posibles son los mensajes *satisfacible*, *no satisfacible* o *desisto*. En el primer caso el algoritmo retorna también una atribución de verdad que satisface la fórmula proposicional de entrada. En el presente trabajo presentamos algunos resultados mostrando que el referido algoritmo puede ser utilizado para resolver instancias arbitrarias del problema de SATISFACIBILIDAD, independientemente de su pertenencia a PURL.

A partir de la clase de satisfacibilidad polinomial PURL se define una jerarquía de clases: $\text{PURL} \subset 2\text{PURL} \subset \cdots \subset k\text{PURL}$, siguiendo la línea de otras jerarquías anteriormente construidas [GS88, DE92, pre96].

El origen de los resultados que mostraremos en el presente texto son los trabajos de investigación desarrollados por el autor en el ámbito del programa de doctorado en Matemática Discreta de la Universidad de Sevilla con el objetivo de resolver los problemas de emparejamiento ortogonal simple en el cilindro (EOSC) y en el toro (EOST).

Un dibujo ortogonal se define de la siguiente manera [BET99]:
Cada vértice está representado por una caja en el espacio $\mathbb{R}^n$ con lados paralelos a los ejes coordenados y cada arista por una secuencia de segmentos conectando las cajas correspondientes a sus extremos. Cada segmento tiene que ser paralelo a uno de los ejes coordenados. El punto donde una arista cambia de dirección se llama *codo*. Si un dibujo ortogonal no contiene codos se llama *rectilíneo*.

La Figura 1 muestra un ejemplo de un dibujo ortogonal donde las cajas rectangulares han sido ya substituidas por los símbolos adecuados a la aplicación práctica. El objetivo principal de los dibujos ortogonales de grafos es obtener dibujos *buenos*. Este calificativo no está definido, a menos que fijemos unos criterios para ello, como el tamaño de la malla, el numero de codos, el numero de cruces o el tamaño de los vértices. Estos cuatro elementos, junto con el espacio en el que consideremos la representación, conducen a diferentes problemas sobre dibujos ortogonales.

En general, en los tipos de representaciones ortogonales considerados en la literatura, cada vértice está asociado a un punto del espacio,

aunque también podemos encontrar trabajos con cajas representando los vértices [EG94, bie95]. Nuestro trabajo utiliza el primer punto de vista.

Una aproximación al diseño VLSI (*Very Large Scale Integration*) de circuitos mediante trazados ortogonales consiste en considerar el circuito a diseñar como un conjunto de pares de vértices que deben unirse mediante aristas, con similares restricciones al problema general del trazado ortogonal de grafos, i.e., es conveniente minimizar el numero de codos y el numero de cruces.

En este sentido se plantea el problema EMPAREJAMIENTO ORTOGONAL SIMPLES PLANO (EOSP), en el cual parejas de puntos en el plano deben unirse mediante trazados ortogonales con un único codo sin que éstos se crucen.

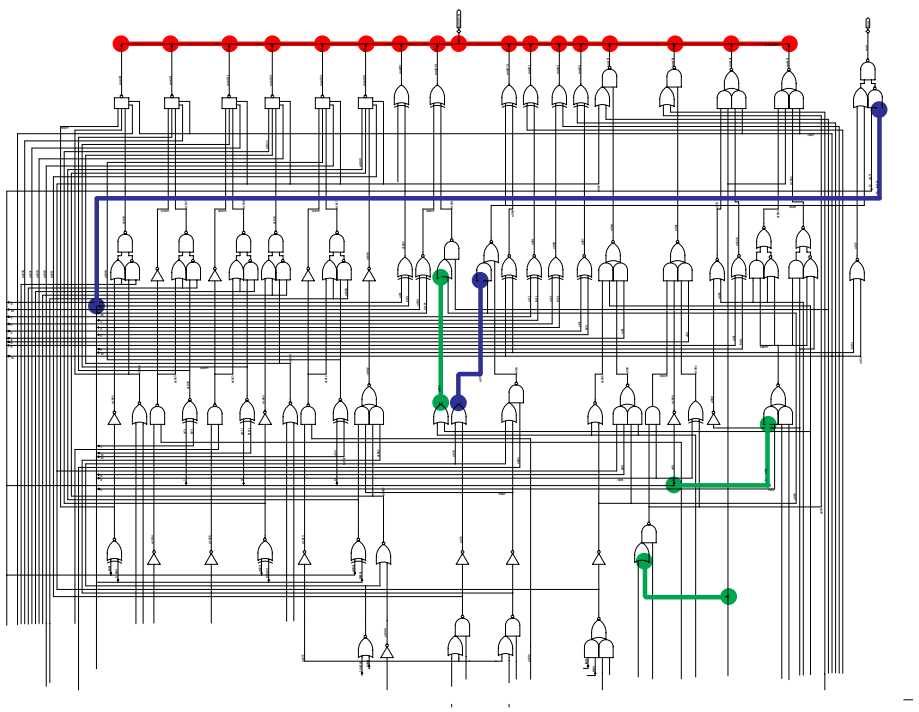


Figura 1: Conexiones ortogonales en un circuito VLSI.

El problema de EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL PLANO (*Single Bend Wiring*) fue estudiado y resuelto por Raghavan, Cohoon y Sahni [RCS86]. Si se codifican los dos posibles caminos para unir cada par de puntos de una entrada del problema mediante una variable booleana, cada instancia del problema geométrico se reduce en tiempo polinomial a una instancia del problema de SATISFACIBILIDAD cuyas cláusulas tienen, como máximo, dos literales (2SAT). Dado que las

instancias de la clase 2SAT pueden ser resueltas en tiempo polinomial, el problema de decisión sobre la existencia (o no) de un emparejamiento ortogonal simple sin que dos trazados se crucen puede ser resuelto también en tiempo polinomial. Para las instancias con respuesta negativa, determinar el mayor número de pares de puntos para los cuales existe un emparejamiento ortogonal sin cruces es un problema NP-duro.

La imposibilidad de consentir cruces entre dos líneas de un circuito integrado lleva a la construcción de *asas* en ésta (v. Figura 2). Ello nos muestra que, en general, el modelado de un circuito no se hace sobre el plano, sino otra superficie. Es decir, el recurso a otras superficies surge como una extensión natural para el estudio de los problemas de emparejamiento ortogonal.

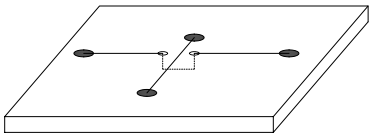


Figura 2: Esquema de una asa en un circuito integrado.

En este sentido, si consideramos un conjunto de pares de puntos, Garrido, Márquez, Morgana y Portillo demostraron que el problema de determinar el menor número g, tal que exista una superficie de género g sobre la que podamos encontrar un emparejamiento ortogonal simple sin cruces para los puntos dados (EOS) es un problema NP-duro.

Una superficie es orientable de género g si es homeomorfa a una esfera con g asas.

Respecto al problema de decisión, dado un conjunto de pares de puntos y una superficie orientable S_g de género g, determinar si en S_g existe un emparejamiento ortogonal simple sin cruces es un problema NP-completo [GMM02]. Quedó como problema abierto establecer la complejidad de cada problema de emparejamiento ortogonal en superficies particulares.

Además del plano, el cilindro, la esfera y el toro son las superficies orientables más simples sobre las que se puede plantear el problema de emparejamiento ortogonal simple. Hay diferencias notables entre estas superficies: para unir un par de puntos con un trazado ortogonal con un codo en el plano existen dos posibilidades, en el cilindro existen cuatro y ocho en la esfera y en el toro. En el presente trabajo consideramos únicamente las superficies cilindro y toro. Las distintas posibilidades para el trazado de una arista ortogonal entre dos vértices en cada una de estas superficies se puede codificar utilizando dos o tres variables booleanas, respectivamente, y así reducir instancias del problema de EMPAREJA-

MIENTO ORTOGONAL SIMPLE EN EL CILINDRO (EOSC) y del problema de EMPAREJAMIENTO ORTOGONAL SIMPLE EN EN EL TORO (EOST) a fórmulas proposicionales. Dicha reducción se puede hacer en tiempo polinomial respecto al número de pares de puntos a unir. Se obtienen de este modo dos clases de satisfacibilidad proposicional, EOSC-SAT y EOST-SAT, que son subclases de 3SAT y 4SAT, respectivamente. Recuérdese que 3SAT y 4SAT son clases de satisfacibilidad NP-completas cuyas formulas están formadas por cláusulas con 3 o 4 literales por cláusula, respectivamente.

Admitiendo que NP$\neq$P, y como el problema general de emparejamiento ortogonal simple es NP-completo [GMM02] y el problema de emparejamiento ortogonal simple en el plano es polinomial [RCS86], EOSC es considerado un problema de frontera, en el sentido de que no se conocía si pertenecía a P o a NP–P. Una de las contribuciones en la presente memoria es la resolución de este problema geométrico. En el Capítulo 4 del presente trabajo, mostramos que EOSC-SAT es una subclase de 2PURL, por lo que el problema EOSC puede ser resuelto en tiempo polinomial.

Figura 3: Extracto de un mapa de carreteras de Portugal (Via Michelín).

En la elaboración de mapas en cartografía, además de la representación de entidades del espacio físico (o político) por medio de elementos gráficos simbólicos, es necesario, incluso imprescindible, el posiciona-

miento de texto informativo: topónimos, identificación de una carretera, río, lago, bosque, etc (v. Figura 3). Escoger el tamaño del texto y el tipo de fuente a utilizar para asegurar la legibilidad y realzar los distintos elementos del mapa y posicionar los rótulos en el mapa evitando ambigüedades son algunos de las decisiones que un cartógrafo precisa tomar durante la larga y compleja tarea de etiquetar un mapa. Uno de los primeros textos en el que se establecen criterios y una metodología para la inserción de texto en un mapa fue elaborado y publicado por el Service Géographique de l'Armée (Paris, 1934) [sgl34]. Posteriormente, en 1975, Imhof presentó un conjunto de criterios, fundamentalmente estéticos, en un trabajo que incluye un amplio conjunto de ejemplos con buenas y malas soluciones en la colocación de etiquetas en un mapa [imh75].

Con el desarrollo y la utilización de procesos automáticos en la producción de cartografía digital, con amplia aplicación en Sistemas de Información Geográfica (SIG), surge la necesidad de algoritmos eficientes para la inserción de etiquetas. El interés de la comunidad científica por este problema concreto es evidente dado el número de trabajos producidos anualmente sobre el tema[9] (v. Figura 4) [WS96].

La Computational Geometry Impact Task Force reconoce también la importancia del tema en su informe de 1999 [CET99]. En éste se señala a los SIG como una de las diez áreas de investigación más importantes en el campo de la Geometría Computacional, identificando el etiquetado de mapas como uno de los problemas más importantes a resolver.

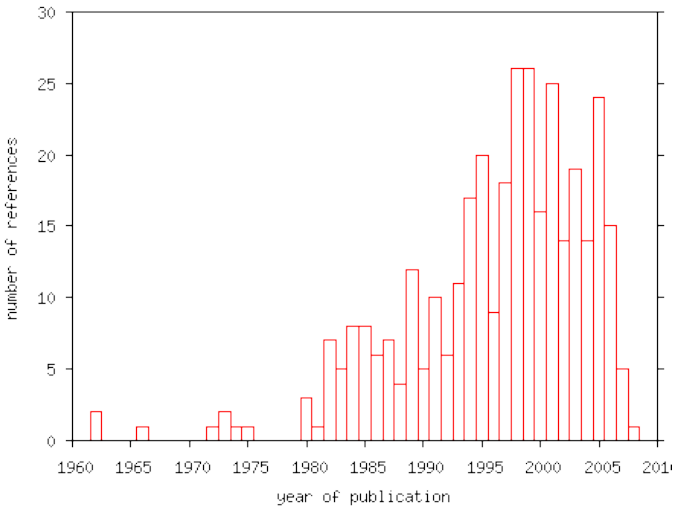


Figura 4: Distribución del número de publicaciones sobre el problema de etiquetado a lo largo del tiempo.

[9] http://i11www.iti.uni-karlsruhe.de/map-labeling/bibliography/

Es obvio que el problema de colocación de texto junto a objetos en una representación gráfica tiene otras aplicaciones además de la cartografía. En general, dado un conjunto de objetos en una representación gráfica y para cada uno de ellos una etiqueta (que puede ser de texto, pero no necesariamente), resolver el problema de etiquetado consiste en encontrar una posición para cada etiqueta, próxima al objeto correspondiente sin que haya solapamiento entre etiquetas. Debido a la dificultad del estudio de problema general se consideran diversos casos particulares. Estos se clasifican de acuerdo a distintos criterios como por ejemplo: según el tipo de los objetos a etiquetar (puntos, segmentos, áreas, conjuntos de puntos, etc), según las características de las etiquetas (forma, tamaño, área, etc) o según el posicionamiento de las etiquetas respecto a los objetos a etiquetar.

En el Capítulo 3 de la presente memoria estudiamos el problema de etiquetado de un conjunto de puntos, alineados según una recta de pendiente unitaria. Las etiquetas se representan por cuadrados con los lados paralelos a los ejes coordenados, que se posicionarán en una de las cuatro posiciones con uno de sus vértices coincidentes con el correspondiente punto a etiquetar (UC-4P). Este problema de etiquetado es idéntico al estudiado por Reyes [rey02] difiriendo apenas en la forma de las etiquetas. En este trabajo, considerando las etiquetas representadas por rectángulos, el referido autor presenta un algoritmo pseudo-polinomial que resuelve el problema en función del número de distintos tamaños de las etiquetas. Es decir, el algoritmo es polinomial si el número de distintos tamaños de las etiquetas está acotado. En el caso particular de que las etiquetas tuvieran todas tamaños distintos el algoritmo propuesto necesita tiempo exponencial para su ejecución. Para el estudio del problema de decisión, relativo a la existencia de solución del problema UC-4P, las cuatro posibles posiciones de cada etiqueta son codificadas por un par de variables booleanas por lo que las instancias de este problema se pueden reducir a instancias de SAT (4SAT más propiamente).

En el presente trabajo se ha estudiado y resuelto el problema de etiquetado UC-4P, presentado en el Capítulo 3. En este Capítulo, demostramos que la clase de satisfacibilidad UC-4P-SAT es subclase de la nueva clase polinomial PURL y que, por lo tanto, el problema de decisión puede ser resuelto en tiempo polinomial.

La presente memoria está formada por cinco capítulos. Se comienza presentando un extenso conjunto de conceptos previos necesarios: complejidad computacional, problema de satisfacibilidad proposicional y clases polinomiales de satisfacibilidad, superficies, emparejamientos ortogonales y problemas de etiquetado. Desde una perspectiva meramente introductoria, pero indispensable para la presentación posterior de cada uno de los diversos temas tratados. En el Capítulo 2 se define el concepto de literal p-eliminable y se caracteriza la nueva clase

de satisfacibilidad proposicional resoluble en tiempo polinomial, PURL. Comparándola con las clases de satisfacción polinomial resolubles en tiempo polinomial más conocidas constatamos que PURL las contiene a todas. Se desarrollan distintos algoritmos polinomiales de complejidad creciente y se define la jerarquía de clases polinomiales, PURL $\subset$ 2PURL $\subset \cdots \subset$ kPURL.

En los dos capítulos siguientes presentamos la resolución de dos problemas que estaban abiertos, ambos de naturaleza geométrica. Las instancias de cada un de los problemas se reducen al problema de SATISFACIBILIDAD PROPOSICIONAL demostrándose que las respectivas clases de satisfacibilidad son subclases de alguna de las clases polinomiales de la nueva jerarquía. En particular, en el Capítulo 3 mostramos que UC-4P-SAT es una subclase de PURL y, en el Capítulo 4, que EOSC-SAT es una subclase de 2PURL. En la demostración de los resultados se usa también el concepto de grafo de interacciones que, traduciendo algunas de las estructuras geométricas de los problemas, contribuye a su resolución.

Además de los resultados presentados en esta memoria, puede encontrarse una implementación del algoritmo ELIMINALITERALES en el Proyecto de Fin de Carrera (master thesis) de Gutiérrez [gut07]. Otras implementaciones de algunos de los algoritmos presentados en esta memoria, implementados en *delphi* y en *python*, se pueden encontrar en `http://w3.ualg.pt/~jirodrig/phd.thesis/`.

Para finalizar se muestran algunas de las principales conclusiones y perspectivas de trabajos futuros. Trabajos en la línea de los resultados obtenidos y, principalmente, en lo mucho que queda por hacer, bien en problemas conocidos que quedan sin resolver, bien en nuevas cuestiones y problemas que ahora aparecen referidos a PURL y a las prometedoras aplicaciones del concepto de literal p-eliminable.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

CAPÍTULO 1

Preliminares

En este capítulo presentaremos las definiciones y conceptos previos que se utilizarán a lo largo del presente trabajo. Fijaremos también las notaciones a utilizar.

Los temas y problemas abordados están íntimamente ligados a las disciplinas de Algorítmica, Lógica Computacional, Teoría de Grafos y Geometría Computacional. Las técnicas de resolución de los problemas abordados tienen aplicación directa en problemas de Geometría Computacional.

Aunque intentemos definir los conceptos más elementales a utilizar posteriormente, es imposible en esta introducción tener la pretensión de presentar todo el edificio matemático necesario para su completa fundamentación, ni siquiera la de esbozarlo. Por eso, para el lector menos familiarizado con estas materias podrá ser útil la consulta de algunas obras obras básicas fundamentales.

Entre ellas podemos destacar el libro de Aho, Hopcroft y Ullman [AHU74] en el dominio de la Algorítmica y la imprescindible obra de Garey y Johnson [GJ79] sobre Complejidad Computacional. En el campo de las aplicaciones de la Algorítmica a la Teoría de Grafos pueden ser consultadas las obras de Nishizeki y Chiba [NC88], y la de Chartrand y Lesniak [CL86].

En el campo de la Teoría de Grafos son ya consideradas obras clásicas de referencia los libros de Harary [har72] y de Bollobas [bol79]. Las obras de introducción a la Lógica Matemática son numerosas y fáciles de encontrar. Por su proximidad a la Teoría de la Computación, el manual de Boolos y Jeffrey [BJ74] es una buena opción, incluso teniendo en consideración alguna desactualización eventual. En el estudio de la relación de los problemas de Satisfacibilidad con la Teoría de

la Complejidad Computacional, el artículo de Cook [coo71] sigue siendo el trabajo de referencia. También sobre Satisfacibilidad, los artículos de Hayes [hay97] o de Schaefer [sch78] son buenas referencias de consulta así como el artículo de Franco y van Gelder [FG03] sobre clases de satisfacibilidad resolubles en tiempo polinomial. La obra de Raffalli [raf95] constituye una buena introducción a los algoritmos utilizados en Lógica Computacional. En el dominio de los problemas de Satisfacibilidad, los libros de Biere [BHM09], de Saïs [sai08] y de Herbstritt [her09] son también obras actuales con un gran interés.

Como a lo largo de este trabajo tratamos problemas de inmersión ortogonal de grafos y, por tanto, de su representación gráfica, es necesario referirnos aquí también a los trabajos de White [whi73], Liu [liu95] así como al libro de Battista, Eades, Tamassia y Tollis [BET99].

Otra disciplina en el campo de los trabajos desarrollados es la Geometría Computacional. Como obras de referencia podemos citar los libros de Preparata y Shamos [PS85], de O'Rourke [oro94] o de Berg, van Kreveld, Overmars y Schwarzkopf [BKO00] así como el interesante libro de Grima y Márquez [GM01] sobre Geometría Computacional en superficies.

1.1. Teoría de Grafos

En general, se utilizará la notación clásica de la Teoría de Grafos, como por ejemplo la adoptada en los libros de Harary [har72] y Bollobas [bol79]. Cuando eso no fuese posible se mencionará explícitamente.

§ 1.1.1. Nociones básicas y definiciones.— Un *grafo simple* (no dirigido) está definido por un par de conjuntos, $G = (V, A)$, donde los elementos de $V = \{v_1, \ldots, v_n\}$ son denominados *vértices* y los elementos de A, pares no ordenados de vértices, se denominan *aristas*. Si $|V| = n$, el grafo se dice que es de orden n. Dos vértices, $v_i, v_j \in V$ se llaman *adyacentes* si la arista $\{v_i, v_j\} \in A$. Dos aristas son *incidentes* si tienen un vértice en común. Adicionalmente, consideramos que si $a \in A$, entonces $a = \{v_i, v_j\}$, con $v_i \neq v_j$, es decir, en un grafo consideramos que no hay aristas de un vértice a sí mismo. Por brevedad, denominaremos *grafos* a los grafos simples siempre que no haya posibilidad de confusión.

Se denomina *grado* o *valencia* de un vértice, $\delta(v)$, al número de vértices adyacentes a v, o equivalentemente, al número de aristas incidentes en v. Un grafo en el cual todos sus vértices tienen la misma valencia, k, se llama $k-$regular. Si fuese de orden n y $(n-1)-$regular se denomina

completo y se denota por K_n. La Figura 1.1 presenta una representación gráfica de dos grafos completos, uno de orden 3 y otro de orden 5.

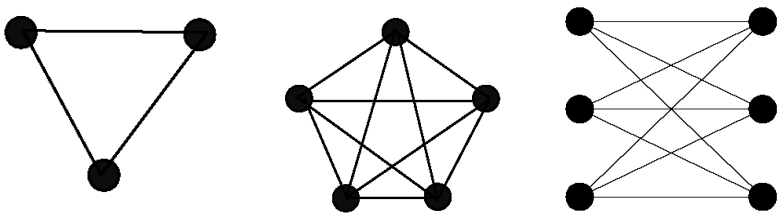


Figura 1.1: Grafos completos K_3 y K_5 y bipartito completo $K_{3,3}$.

Un grafo se llama *bipartito* si existe una partición del conjunto de los vértices $V = V_1 \cup V_2$, ($V_1 \cap V_2 = \emptyset$) de modo que, para cualquier arista, $\{v_i, v_j\} \in A$, $v_i \in V_1$ y $v_j \in V_2$ o viceversa.

Dados dos vértices v_i, v_j de un grafo G, un *camino* en G es una lista ordenada de vértices $P = \{v_i, w_1, \cdots, w_k, v_j\}$, de modo que cada vértice es adyacente al anterior y al siguiente. Un camino se denomina simple si no se repiten vértices. Si existe un camino entre dos vértices, entonces también existe un camino simple entre esos mismos dos vértices. De modo equivalente, un camino puede ser definido por una secuencia ordenada de aristas formadas por vértices consecutivos, $\{\{v_i, w_1\}, \{w_1, w_2\}, \ldots, \{w_k, v_j\}\}$. Se denomina *longitud* de un camino al número de aristas que lo componen. En el ejemplo anterior, la longitud de P es igual a $k+1$; lo que corresponde al número de vértices de la lista ordenada ($k+2$) menos uno. Un camino comenzando y terminando en el mismo vértice se denomina *circuito*. El circuito se denomina *ciclo* si ese es el único vértice repetido.

Un grafo no vacío, G, se llama *conexo* si para cualesquiera dos vértices de G, existe un camino entre ellos. Podemos definir una relación $\mathcal{R}$ entre los vértices de G del siguiente modo: $v\mathcal{R}w$ si existe un camino entre v y w. $\mathcal{R}$ es una relación de equivalencia, pues es fácil concluir que es reflexiva, simétrica y transitiva. Llamaremos componentes conexas del grafo G a las clases de equivalencia del conjunto $G/\mathcal{R}$.

Se dice que $G' = (V', A')$ es un *subgrafo* de G si $V' \subseteq V$ y $A' \subseteq A$. Dado un subconjunto de vértices $W \subseteq V$, se llama *subgrafo inducido*, $G(W)$, al subgrafo de G tal que el conjunto de los vértices de $G(W)$ es W y las aristas de $G(W)$ son las aristas de G cuyos vértices pertenecen a W.

Un vértice v es denominado *vértice de corte* si el número de componentes conexas del subgrafo inducido $G(V - \{v\})$ aumenta con respecto a G. Un grafo sin vértices de corte se denomina *2-conexo*.

§ 1.1.2. Grafo cordal o triangular.— A continuación presentamos la definición y una caracterización de los grafos triangulares. Este tipo de grafos tiene un interés especial en el estudio del problema EOSC que investigaremos en el Capítulo 4.

Dado un grafo y un ciclo C, una arista $\{v_i, v_j\}$ se llama *cuerda* si los vértices $v_i, v_j \in C$ pero no son consecutivos. Un grafo se llama cordal o triangular si todos los ciclos de longitud superior a tres contienen una cuerda.

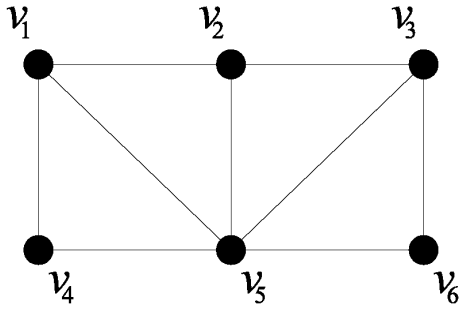


Figura 1.2: Grafo triangular.

Para un vértice $v \in V$, el conjunto de los vértices adyacentes a v se llama se llama *entorno* de v y se denota por $N(v) = \{w \in V : \{v, w\} \in A\}$. Un vértice $v \in V$ se dice *simplicial* si el subgrafo inducido $G\left(N(v) \cup \{v\}\right)$ es un grafo completo. Los grafos triangulares pueden ser caracterizados a partir de los vértices simpliciales. Si G es un grafo triangular, pero no completo, entonces contiene al menos dos vértices simpliciales no adyacentes [dir61]. Siendo G un grafo triangular y v_k un vértice simplicial, el subgrafo inducido $G\left(V - \{v_k\}\right)$ es también un grafo triangular. Procediendo de igual modo con los sucesivos subgrafos, eliminando en cada paso un vértice simplicial del subgrafo obtenido se llega a una secuencia de vértices $\{v_{i_1}, \ldots, v_{i_n}\}$, llamada *secuencia de eliminación perfecta*. G es un grafo triangular si y sólo si admite una ordenación de eliminación perfecta para sus vértices [FG65].

La Figura 1.2 muestra un ejemplo de grafo triangular, en el cual $\{v_4, v_1, v_2, v_6, v_3, v_5\}$ es una secuencia eliminación perfecta.

Resultados sobre grafos triangulares pueden ser encontrados en diversos trabajos recientes: [HHM04, SS03, CIR03].

1.2. Complejidad computacional

§ 1.2.1. Problemas, algoritmos y complejidad.— En Ciencias de la Computación, un *problema* se caracteriza mediante un par formado por un conjunto de datos o parámetros, a los que llamamos *entrada*, y por una pregunta acerca de una propiedad o característica sobre los datos de la entrada. Un *algoritmo* es un método de obtención de la respuesta a la pregunta formulada.

Un poco a semejanza de las clasificaciones que se dan en los juegos y puzzles relacionándolos con las edades de los niños como indicador del grado de dificultad, los algoritmos son clasificados respecto a su nivel de *complejidad*. La complejidad de un algoritmo corresponde a una medida de la dificultad de su ejecución.

El estudio de la complejidad se puede hacer mediante distintos aspectos que expresan la cantidad de un determinado recurso requerido por el algoritmo. En el caso de algoritmos que deben ser ejecutados en un computador, el estudio de la complejidad se centra principalmente en el tiempo, cantidad de memoria o número de procesadores. En este trabajo consideraremos tan sólo el estudio de la complejidad de los algoritmos computacionales desde la perspectiva del tiempo que tarda su ejecución. Para una aproximación al estudio de la complejidad espacial, la cantidad de recursos de memoria requeridos por un algoritmo, el libro de Garey y Johnson [GJ79] y el trabajo de Meyer y Shamos [MS77], constituyen un buen punto de partida.

Para el estudio de la *complejidad temporal*, consideraremos el modelo computacional presentado y seguido en el libro de Garey y Johnson [GJ79] y la notación adoptada en esta obra así como en los trabajos de Knuth [knu76] y de Cook [coo71]. De un modo general, estas referencias constituyen también una buena introducción a la Teoría de la Complejidad Computacional.

El tiempo utilizado por un computador depende, fundamentalmente, de sus propias características técnicas, del tamaño de la entrada y del algoritmo implementado. Para que la medida sea independiente de la implementación, en Ciencias de la Computación esa medida es cuantificada por el número de *operaciones activas* en función del tamaño de la entrada n y se representa por $f(n)$.

Para su representación adoptaremos la *notación asintótica* introducida en 1894 por Bachmann [bac94] y popularizada desde su incorporación sistemática por Knuth [knu76]. Todos los análisis se refieren a la complejidad computacional correspondiente al peor caso que puede presentarse. Se denota $O(g(n))$ el conjunto de todas las funciones $f(n)$ tales que existen una constante positiva C y un número natural n_0 de forma que $|f(n)| \leq Cg(n)$ para todo $n \geq n_0$.

Por ejemplo, tomemos un problema que tenga como entrada un conjunto de n números enteros y como pregunta *¿Cual es el menor entre ellos?* Un algoritmo que tome cada uno de los valores de la entrada y los compare con todos los restantes tendrá $f(n) = n(n-1)$ operaciones. En realidad, cada comparación es precedida por una operación de lectura, pero, como veremos más adelante, la complejidad no se altera por el hecho de que cada operación activa esté constituida por k operaciones, siempre que k no dependa de n. En este ejemplo, diremos que el algoritmo tiene complejidad de orden n cuadrado y la denotamos $O(n^2)$.

Si consideramos un problema con la misma pregunta, pero en el en que las entradas estén constituidas por secuencias crecientemente (resp. decrecientemente) ordenadas de n números enteros, el menor entero puede ser identificado tomando simplemente el primer (resp. el último) elemento de la secuencia. Independientemente de la dimensión de la entrada, la respuesta se obtiene en tiempo constante. Diremos entonces que la complejidad de este algoritmo es $O(1)$.

Un algoritmo se llama *polinomial* si su complejidad temporal es $O(n^p)$, para algún $p \in \mathbb{N}$ (independiente de n), y exponencial en los restantes casos. En particular, un algoritmo se dice *cuadrático* si $p = 2$, *lineal* si $p = 1$ o que *corre en tiempo constante* si $p = 0$. Esta distinción asume particular significado para entradas con elevados valores de n, como se puede observar en el Cuadro 1.1. Este cuadro presenta los tiempos de ejecución en segundos, para distintos valores de n y diferentes complejidades en un computador que efectúe 2^{20} operaciones activas por segundo. Los datos del cuadro permiten estudiar la variación del tiempo de procesamiento según el tamaño de las entradas (n) y la complejidad (n, n^2, n^3, n^5, 2^n y 3^n). Se puede comprobar que los algoritmos polinomiales presentan un tiempo de procesamiento razonable, mientras que los exponenciales apenas permiten la resolución (en tiempo útil) de problemas pequeños.

Cuadro 1.1: Tiempos de computación.

	n					
$f(n)$	10	30	50	100	500	1000
n	10 μs	29 μs	48 μs	95 μs	477 μs	954 μs
n^2	0.1 ms	0.9 ms	2.4 ms	9.5 ms	0.2 s	1.0 s
n^3	1.0 ms	25.7 ms	119.2 ms	1.0 s	2.0 s	15.9 s
n^5	95.4 ms	23.2 s	5.0 m	2.6 h	11.5 mm	30.7 aa
2^n	1.0 ms	17.1 m	34.0 aa	$3.8 \cdot 10^{14}$ sc	$---$	$---$
3^n	0.1 s	75.8 mm	$2.2 \cdot 10^8$ sc	$---$	$---$	$---$

$(*)$ $s-$segundo, $1\mu s = 10^{-6}s$, $1ms = 10^{-3}s$, $m-$minuto, $h-$hora, $d-$día, $mm-$més, $aa-$año, $sc-$siglo

Por el modo en que se definen, tienen la misma complejidad, por ejemplo $O(n \cdot log(n))$, un algoritmo que ejecute $100n \cdot log(n)$ o $\frac{n \cdot log(n)}{100}$ operaciones activas. El Cuadro 1.2 permite comprobar que los algoritmos polinomiales aunque contengan una constante de multiplicación grande, siguen siendo preferibles a los exponenciales.

Cuadro 1.2: Tiempos de computación.

	n			
$f(n)$	10	100	500	1000
$1000n$	$9.5\ ms$	$95.4\ ms$	$476.8\ ms$	$953.7\ ms$
$1000n \log n$	$31.7\ ms$	$633.6\ ms$	$4.3\ s$	$9.5\ s$
$1000n^3$	$1.0\ s$	$15.9\ m$	$1.4\ dd$	$11.0\ dd$
2^n	$1.0\ ms$	$3.8 \cdot 10^{14}\ sc$	$---$	$---$

$(*)$ $s-$segundo, $1\mu s = 10^{-6}s$, $1ms = 10^{-3}s$, $m-$minuto, $h-$hora, $d-$día, $mm-$més, $aa-$año, $sc-$siglo

La evolución tecnológica permite la producción de computadores cada vez más rápidos. El Cuadro 1.3 presenta el tamaño de la mayor entrada para la cual la solución del problema es obtenida en una hora de procesamiento, para diferentes complejidades y diferentes computadores. La segunda columna indica la dimensión de la mayor entrada para la cual la solución se obtiene en una hora de procesamiento para un cierto computador. La tercera y cuarta columnas representan la dimensión de la máxima entrada resoluble en una hora de procesamiento por otros dos computadores, uno cien y el otro mil veces más rápido, respectivamente, que el correspondiente a la segunda columna. El aumento de la velocidad del computador tiene un efecto prácticamente nulo en los algoritmos de complejidad exponencial.

Cuadro 1.3: Tamaño máximo del problema resoluble en una hora.

tiempo	computador	computador 100 veces más rápido	computador 1000 veces más rápido
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5 + 7$	$N_5 + 10$
3^n	N_6	$N_6 + 4$	$N_6 + 6$

Los ejemplos presentados indican algunas de las razones por las cuales los algoritmos polinomiales son preferibles a los exponenciales. La distinción entre estos dos grupos de algoritmos data de 1964, de Edmonds [edm65] y Cobman [cob64]. En particular, Edmond clasificaba los algoritmos polinomiales como buenos algoritmos y conjeturaba la existencia de problemas que no pudieran ser resueltos por tales algoritmos.

En esta línea de clasificación, un problema está bien resuelto si se conoce un algoritmo polinomial que lo resuelva. Un problema para el cual no sea conocido ningún algoritmo polinomial que lo resuelva y que por su dificultad se considere que es posible que tal algoritmo no exista es, de algún modo, un problema intratable.

Aunque el concepto de problema intratable sea impreciso, constituye la motivación para las definiciones de problemas NP, NP-completos e NP-duros a las que nos referiremos posteriormente.

§ 1.2.2. Clases de complejidad: P y NP.— Un problema se llama de *decisión* si el espacio de soluciones contiene dos únicos valores *sí* y *no*. Siendo Π uno de estos problemas, denotamos por D_Π el conjunto de sus entradas y por Y_Π, el subconjunto de sus entradas con respuesta afirmativa.

Este tipo de problemas tiene especial interés pues cualquier problema puede ser reducido a un problema de decisión. Por otro lado, la complejidad de los diversos problemas de decisión puede ser comparada, conduciendo a la definición de las clases de complejidad. Las más habituales son las clases P y NP: Polinomial y No determinista Polinomial, respectivamente.

A la clase P pertenecen todos los problemas de decisión que pueden ser resueltos por un algoritmo determinista de complejidad polinomial. Informalmente, un algoritmo se llama *determinista* si para cada entrada del problema proporciona una solución. Para una instancia particular del problema, la secuencia de estados y el resultado de salida son siempre los mismos. En el ámbito de las ciencias de computación, un algoritmo determinista es un algoritmo computacional que tiene un comportamiento previsto, en que cada cambio de estado de los datos es único.

Las definiciones formales de algoritmo, de algoritmo determinista y de la clase P, adoptadas en nuestro trabajo, son las basadas en el modelo computacional inducido por las maquinas de Turing deterministas y se pueden encontrar en el livro de Garey y Johnson [GJ79].

Para definir la otra de las clases referidas, NP, precisamos introducir el concepto de algoritmo no determinista. Estos algoritmos funcionan en dos etapas, la de conjetura (*guess*) y la de verificación (*check*). En la etapa de conjetura se produce una estructura relacionada con la solución

para la cual se comprueba en la fase de verificación si corresponde a una respuesta afirmativa. Un algoritmo no determinista se denomina polinomial si la etapa de verificación puede ser realizada en tiempo polinomial. Así, un problema de decisión pertenece a la clase NP si puede ser resuelto por un algoritmo no determinista polinomial.

Si Π es un problema en P entonces existe un algoritmo polinomial que lo resuelve y, por lo tanto, existe también un algoritmo no determinista polinomial que lo resuelve. Como consecuencia, Π está en NP. Es decir, P $\subseteq$ NP. Está por demostrar si P $\neq$ NP, aunque se conjetura que sí.

Bajo la definición de NP, si admitimos que P $\neq$ NP, los problemas de decisión que pertenezcan a NP–P son problemas intratables.

Dados dos problemas de decisión, Π_1 y Π_2, se dice que $f : D_{\Pi_1} \longrightarrow D_{\Pi_2}$ es una transformación o reducción polinomial del problema Π_1 al problema Π_2, ($\Pi_1 \propto \Pi_2$), si existe un algoritmo determinista polinomial que haga la transformación de las entradas del problema Π_1 en el problema Π_2 en tiempo polinomial, de modo que, para cada entrada $I \in D_{\Pi_1}$, I tiene respuesta afirmativa ($I \in Y_{\Pi_1}$) si y sólo si $f(I)$ tiene también respuesta afirmativa ($f(I) \in Y_{\Pi_2}$). En estas condiciones, si existe un algoritmo polinomial que resuelva el problema Π_2 también resuelve el problema Π_1. De igual modo, si Π_1 es un problema intratable, Π_2 también lo es.

El problema de decisión Π es NP-completo si Π es NP y para cualquier problema Π', NP, existe una reducción polinomial del problema Π' a Π. Con esta construcción, si existe un algoritmo polinomial que resuelva un problema NP-completo entonces todos los problemas NP pueden ser resueltos en tiempo polinomial. Por otro lado, si alguno es intratable, todos son intratables. Admitiendo que P$\neq$NP, la Figura 1.3 presenta una versión del mundo NP.

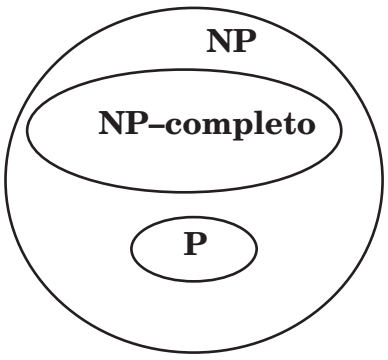


Figura 1.3: Una versión del mundo de la complejidad computacional.

Hemos visto, por tanto, que un problema Π es NP-completo si verifica las siguientes propiedades:

- Π es NP;
- Para cualquier problema NP, $\Pi^{'}$, entonces $\Pi^{'} \propto \Pi$.

Teorema 1.1 (Teorema de Cook [coo71])**.** *El problema de Satisfacibilidad es un problema NP-completo.*

Por tanto, si P≠NP, los problemas de decisión intratables (problemas en NP–P) son problemas NP-completos y son al menos tan difíciles de resolver como el problema de Satisfacibilidad. Para demostrar que un dado problema Π es NP-completo, se puede seguir el siguiente procedimiento, equivalente a la definición:

1. Demostrar que Π pertenece a la clase NP.
2. Seleccionar un problema conocido, $\pi^{'}$, NP-completo.
3. Construir una transformación, f, de $\pi^{'}$ a π.
4. Probar que f es una transformación polinomial.

§ 1.2.3. Problemas NP–duros.— Hasta ahora nuestro estudio se ha centrado en la clase de problemas NP, sin embargo pueden existir problemas fuera de esta clase que sean *tan duros* como los NP-completos. A ellos dedicamos el concepto de problema *NP-duro*.

Así, diremos que un problema es NP-duro si todos lo problemas en NP son reducibles a él en tiempo polinomial y diremos que es NP-completo si es NP-duro y está en NP.

De forma equivalente a la definición anterior, un problema Ω es un problema NP-duro si existe un problema Π, NP-completo, reducible a Ω en tiempo polinomial ($\Pi \propto_T \Omega$). Por lo tanto, si existe un algoritmo determinista polinomial que resuelva algún problema NP-duro entonces habría algoritmos deterministas polinomiales para todos los problemas en NP y, en consecuencia se tendría que P=NP.

1.3. Problema de satisfacibilidad

Sea $V_n = \{v_1, \ldots, v_n\}$ un conjunto de n *variables lógicas*. A cada variable v_i le corresponden dos *literales*, v_i y $\overline{v_i}$. El primero se denomina literal *afirmado* y el segundo, literal *negado*. El conjunto de literales sobre las variables se denota $L_n = \{v_1, \overline{v_1}, v_2, \overline{v_2}, \ldots, v_n, \overline{v_n}\}$. En general,

siempre que no se exprese lo contrario, los elementos de un subconjunto de literales de L_n serán representados como $l_1, l_2, \ldots, l_t$, sin que el índice i de literal l_i tenga ninguna correspondencia con el literal v_i o $\overline{v_i}$. Cada literal l_i podrá representar un literal afirmado o negado, v_k o $\overline{v_k}$ y en este caso, $\overline{l_i}$ representará su complementario, $\overline{v_k}$ o v_k, respectivamente. Identificaremos siempre $\overline{\overline{l_i}} = l_i$. Para referir que un literal corresponde a una variable determinada, se utilizará el símbolo v_i^* para designar específicamente uno de los literales, v_i o $\overline{v_i}$, sobre la variable v_i.

Una *cláusula* C corresponde a un subconjunto de literales de L_n y por convención se denotará $C = [l_1, \ldots, l_k]$, representando la disyunción de sus literales. A veces también se representa como $l_1 \vee l_2 \vee \cdots \vee l_k$. Una secuencia de cláusulas, $\mathcal{F} = \{C_1, \ldots, C_m\}$, se llama *fórmula proposicional* en forma normal conjuntiva o CNF-fórmula y representa la conjunción de las cláusulas. A veces también se representa como $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. En general se supone que una fórmula proposicional no contiene cláusulas repetidas, pues, en caso contrario, estas podrían ser eliminadas en tiempo polinomial. Dado que las susodichas cláusulas pueden ser ordenadas lexicográficamente en tiempo $O(m \cdot log(m))$, para eliminar las repetidas sería suficiente comparar cada cláusula con la consecutiva eliminándola si fuese igual.

Se denomina tamaño de una fórmula a $|\mathcal{F}| = \sum_{i=1}^{m} |C_i|$, donde $|C_i|$ designa el número de literales de la cláusula. Cuando cada cláusula tiene exactamente k literales, $|\mathcal{F}| = km$.

Se representa por $var(\mathcal{F})$ al conjunto de las variables booleanas para las cuales alguno de sus literales ocurre en una cláusula. Se tiene naturalmente que $var(\mathcal{F}) \subset V_n$.

Una *atribución* de valores de verdad sobre el conjunto de literales, L_n, es una función $t : L \subseteq L_n \rightarrow \{0, 1\}$, tal que $t(v) = 1$ si y sólo si $t(\overline{v}) = 0$, para cada literal $v \in L$ ($\overline{\overline{v}} = v$). Los valores del conjunto $\{0, 1\}$ corresponden a los valores lógicos falso o verdadero, F o V. Si $L \subsetneq L_n$, la atribución de valores de verdad t se llama parcial. Abreviaremos como *atribución de verdad* o simplemente *atribución* o *valoración* a la atribución de valores de verdad. La representaremos como un subconjunto de literales τ, todos con valor 1, es decir, un literal $v \in \tau$ si y sólo si $t(v) = 1$.

Para una atribución de verdad τ, una cláusula asume valor verdadero, $C_i(\tau) = 1$, si alguno de los literales de C_i es verdadero para la atribución τ ($C_i \cap \tau \neq \emptyset$). En caso contrario la cláusula asume valor falso. Una CNF-fórmula $\mathcal{F}$ tiene valor verdadero para la atribución τ, (la atribución τ satisface $\mathcal{F}$) cuando todas las cláusulas de $\mathcal{F}$ son verdaderas para la susodicha atribución y, en ese caso, denotamos $\mathcal{F}(\tau) = 1$. En caso contrario, si existe una cláusula falsa, se dice que la fórmula es falsa y denotamos $\mathcal{F}(\tau) = 0$. Una atribución τ satisfaciendo $\mathcal{F}$ se denomina un *modelo* de $\mathcal{F}$.

Para una atribución de verdad, se dice que una variable booleana v es *verdadera* (o que toma el valor lógico verdadero) si el literal afirmado correspondiente es verdadero. Se dice que la variable v es *falsa* si el literal negado correspondiente es verdadero. En cualquier de los casos se dice que la variable v está atribuida.

Dos fórmulas proposicionales sobre el mismo conjunto de variables se dice que son *lógicamente equivalentes* si, para cada atribución de verdad ambas tienen el mismo valor (ambas son verdaderas o ambas falsas). Se dice que dos fórmulas son *equisatisfacibles* si ambas son satisfacibles o ambas son no satisfacibles.

Una cláusula vacía (sin literales) se representa por el símbolo $\square$ y es falsa para cualquier atribución de verdad. Una fórmula vacía, sin cláusulas, es verdadera para cualquier atribución de verdad.

Con estas definiciones previas, el problema de SATISFACIBILIDAD PROPOSICIONAL se puede enunciar de la siguiente forma:

SATISFACIBILIDAD (SAT) [coo71]
Entrada: una CNF-fórmula proposicional $\mathcal{F}$.
Pregunta: ¿Existe una atribución de valores de verdad que satisfaga $\mathcal{F}$?

El problema de SATISFACIBILIDAD tiene una importancia central en el dominio de la lógica computacional pues fue el primer problema NP-completo conocido. Más aún, todos los problemas de decisión de la clase NP se pueden reducir en tiempo polinomial al problema SAT.

Introduciendo ciertas restricciones al problema SAT se pueden obtener subclases de satisfacibilidad en la clase P.

kSAT constituye un caso particular de clases de SATISFACIBILIDAD en el cual cada cláusula contiene, como mucho, k literales. Las subclases 2SAT y 3SAT constituyen ejemplos paradigmáticos en el estudio de la complejidad. El problema de satisfacibilidad proposicional de las instancias 2SAT es resoluble en tiempo polinomial [EIS76] y el de las instancias 3SAT es un problema NP-completo [coo71].

2-SATISFACIBILIDAD (2SAT)
Entrada: Una CNF-fórmula proposicional $\mathcal{F} = \{C_1, C_2, \ldots, C_m\}$ con $|C_i| = 2, i = 1, \ldots, m$.
Pregunta: ¿Existe una atribución de verdad que satisfaga $\mathcal{F}$?

3-SATISFACIBILIDAD (3SAT)
Entrada: Una CNF-fórmula proposicional $\mathcal{F} = \{C_1, C_2, \ldots, C_m\}$ con $|C_i| = 3, i = 1, \ldots, m$.
Pregunta: ¿Existe una atribución de verdad que satisfaga $\mathcal{F}$?

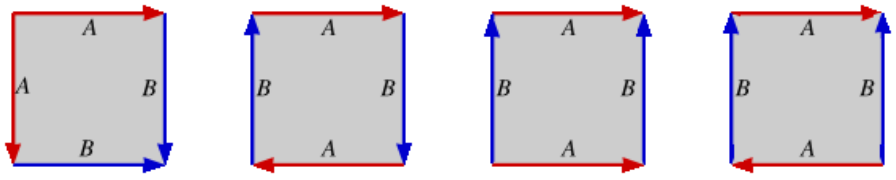
Las subclases de satisfacibilidad son llamadas *clases de satisfacibilidad* y las subclases resolubles por algoritmos deterministas polinomiales reciben el nombre de *clases de satisfacibilidad polinomiales*.

El problema de la SATISFACIBILIDAD de las instancias 3SAT es un problema NP-completo [coo71]. Por lo tanto no se conoce ningún algoritmo determinista polinomial que lo resuelva. Diremos entonces que esta es una clase de satisfacción NP-completa.

1.4. Superficies

Uno de los problemas que estudiaremos en el presente trabajo es el de emparejamiento ortogonal en superficies y, en particular, en el cilindro. A ese efecto, necesitamos fijar algunos conceptos generales sobre superficies. Supondremos conocidos los conceptos topológicos elementales y definiremos una superficie cerrada S como un espacio topológico compacto de Hausdorff, conexo por arcos y localmente homeomorfa a un disco. Nótese que esta definición excluye el plano y el cilindro, por tratarse de espacios no cerrados. Sin embargo, resulta también interesante estudiar trazados ortogonales en estas dos superficies.

Una superficie cerrada puede construirse a partir de un polígono orientado con un número par de lados (polígono fundamental de la superficie), identificando sus lados dos a dos. Como ejemplo consideremos los cuadrados orientados de la Figura 1.4. En cada uno de ello, si hacemos coincidir los lados A con A y B con B de modo que la orientación de las flechas coincida se obtienen cuatro superficies elementales: la esfera (a), el plano proyectivo (b), el toro (c) y la botella de Klein (d).



(a) (b) (c) (d)

Figura 1.4: Polígono fundamental de la esfera (a), del plano proyectivo (b), del toro (c) y de la botella de Klein (d).

Considerando dos superficies, M y N, recortando en cada una de ellas un disco y uniéndolas por las fronteras de los respectivos recortes se obtiene una nueva superficie, designada como *suma conexa* de las dos

primeras, $M\#N$. La Figura 1.5 ilustra la suma conexa del doble toro (S_2) con el toro (S_1). La superficie así obtenida es el triple toro (S_3).

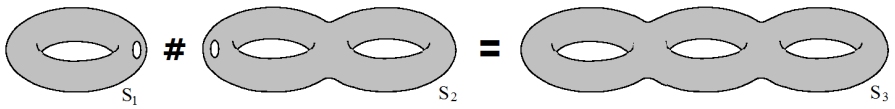


Figura 1.5: Suma conexa del doble toro con el toro.

De acuerdo al Teorema de clasificación de las superficies cerradas, cualquier superficie es homeomorfa a una superficie miembro de una de las siguientes clases [FF67]:

1. esfera;
2. suma conexa de g toros, con $g \geq 1$;
3. suma conexa de k planos proyectivos, con $k \geq 1$.

Existen dos tipos de superficies cerradas, las orientables y las no orientables. De entre las clases anteriores, las dos primeras están formadas por superficies orientables. La esfera se denota por S_0, el toro por S_1 y la suma conexa de g toros por S_g. El valor de g se denomina *género* de la superficie.

§ 1.4.1. Representación ortogonal estándar de superficies.—

Toro

Como vimos anteriormente, el toro es una superficie orientable de género 1. Puede ser obtenida por la rotación de una circunferencia, Q, de radio r, alrededor de una recta cualquier de su plano, situada a una distancia $d > r$ del centro de Q. Las circunferencias definidas por cada uno de los puntos de Q durante la rotación son llamadas *paralelos* y las circunferencias correspondientes a Q en cada una de las posiciones durante la rotación en torno de la recta, son llamadas *meridianos*. Paralelos y meridianos constituyen un sistema ortogonal de líneas coordenadas (v. Figura 1.6).

Cortando el toro por un meridiano y por un paralelo obtenemos una representación en la cual los bordes superior y inferior están identificados, al igual que el izquierdo y el derecho (v. Figura 1.7). En virtud del homeomorfismo existente entre los puntos del toro y los del rectángulo obtenido y por ser una representación del toro sobre el plano, esta construcción recibe la designación de *toro plano*.

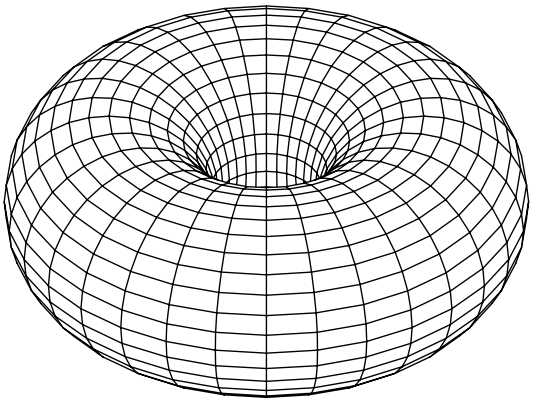


Figura 1.6: Sistema de líneas coordenadas ortogonales en el toro.

En esta representación, las líneas coordenadas son representadas por segmentos paralelos a los lados del toro plano.

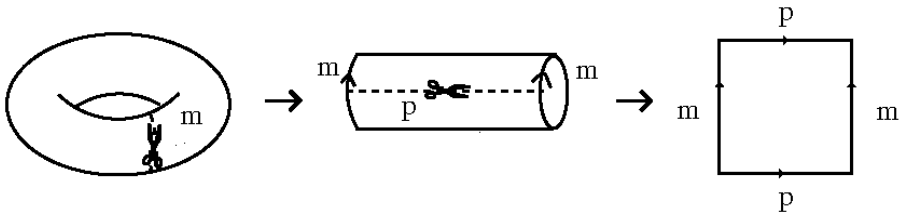


Figura 1.7: Construcción del toro plano.

Superficies orientables de género $g \geq 2$

El toro plano es una representación plana de una superficie cerrada de género $g = 1$, Cobos (1995) obtiene la representación ortogonal estándar de todas las superficies de género $g \geq 2$. El método es el siguiente: si recortamos en el toro plano una ventana, o sea, un rectángulo de lados paralelos a los lados del toro plano, y en ese rectángulo identificamos el lado superior con el inferior y el izquierdo con el derecho, se obtiene una superficie plana homeomorfa al doble toro. La Figura 1.8 explica este proceso: en la izquierda vemos el doble toro y a la derecha su representación ortogonal estándar.

Figura 1.8: Doble toro (izquierda) y la correspondiente representación ortogonal estándar O_2 (derecha).

Abriendo $g-1$ ventanas en la representación ortogonal del toro plano se obtiene una representación plana homeomorfa a una superficie cerrada orientable de género g (v. Figura 1.9).

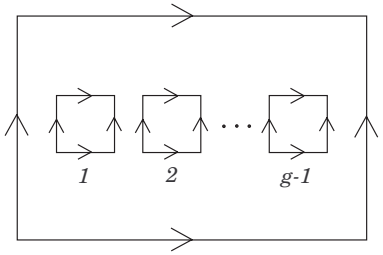


Figura 1.9: Representación ortogonal estándar de una superficie de género g, O_g.

Teorema 1.2. *[cob95] Toda superficie cerrada orientable y conexa es homeomorfa a una de las superficies estándar* $O_g,\ g \geq 0$.

Cilindro

No siendo una superficie cerrada, el cilindro carece de una referencia específica. Esta superficie, orientable y no limitada, se obtiene geométricamente por la rotación de una recta, r en torno de una otra paralela a una distancia $d > 0$. Podemos obtener un sistema de representación del cilindro (análogo al obtenido en el toro), efectuando un corte de esta superficie por una generatriz (meridiano). De esta forma tenemos una banda infinita en el plano con dos lados identificados, como se muestra en la Figura 1.10.

Las rectas correspondientes a r en cada una de las posiciones durante la rotación, son denominadas meridianos y las circunferencias descritas por cada uno de sus puntos, $P \in r$, paralelos. Meridianos y paralelos del cilindro constituyen un sistema de líneas coordenadas ortogonales.

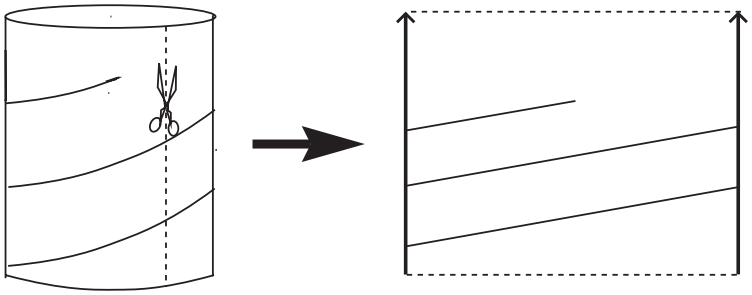


Figura 1.10: Construcción del cilindro plano.

1.5. Emparejamientos ortogonales

§ 1.5.1. Inmersión de grafos en superficies.— Definimos en la Sección 1.1 un grafo como un objeto meramente combinatorio. Sin embargo, con frecuencia es necesario recurrir a su representación topológica, i.e., dibujarlo sobre diversas superficies. En cada una de las representaciones, las aristas pueden ser representadas por diferentes tipos de líneas. En el presente trabajo tienen particular interés las representaciones de las aristas por secuencias de segmentos rectilíneos ortogonales entre sí.

Numerosas aplicaciones requieren la representación de grafos en distintas superficies: en el plano, en el cilindro, en superficies orientables de género g e incluso en superficies no orientables. Es objetivo de esta sección fijar los conceptos de base sobre la inmersión de grafos en superficies particulares, especialmente en el plano, en el cilindro y en el toro. Son de especial interés las definiciones y resultados relacionados con los problemas de emparejamiento ortogonal. Se recomienda al lector interesado la consulta de otras obras sobre el tema, como por ejemplo la excelente introducción incluida en la tesis de doctorado de Portillo [por02]. Es casi obligatoria también la consulta del trabajo de Cobos [cob95].

Se dice que un grafo admite una *inmersión* en una superficie S, si es posible dibujarlo sobre esa superficie representando los vértices por puntos y las aristas por curvas de Jordan de modo que dos aristas no se intersequen en puntos de la superficie que no sean vértices. Considerando un grafo G como un espacio topológico con la estructura de un complejo 1-dimensional, una inmersión de G en una superficie S puede ser considerada como una aplicación continua $\varphi : G \to S$, cuya imagen es homeomorfa a G mediante φ.

Un grafo $G = (V, A)$ se llama *plano* si admite una inmersión en el plano. Para la caracterización de grafos planos se recorda el notable

Teorema de Kuratowski:

Teorema 1.3. *[kur30] Un grafo es plano si y sólo si no contiene ningún subgrafo homeomorfo a K_5 ni a $K_{3,3}$.*

Si G no admite una inmersión en el plano pero la admite en el cilindro, se denomina por *grafo cilíndrico*. Si no admite una inmersión en el cilindro pero la admite en el toro, se llama *grafo tórico*.

§ 1.5.2. Emparejamientos ortogonales.— Según Battista *et al.* (1999) una representación ortogonal de un grafo $G = (V, A)$ en $\mathbb{R}^2$ es un dibujo en el cual los vértices están representados por puntos y cada arista por una secuencia de segmentos. Cada segmento es paralelo a uno de los ejes coordenados y el punto dónde la arista cambia de dirección es llamado *codo* [BET99].

Con el fin de fijar la notación, denominaremos *arista ortogonal* al trazado ortogonal de una arista en las condiciones anteriores.

En la Figura 1.11 se presenta un ejemplo con tres pares de puntos en el plano y, para cada par, un arista ortogonal minimizando el número total de codos. Obsérvese que las aristas no se cruzan. La arista ortogonal que une el par $w_1 = (a_1, b_1)$ no tiene codos, la arista que une el par $w_2 = (a_2, b_2)$ tiene un codo y la arista que une los puntos del par $w_3 = (a_3, b_3)$, dos codos.

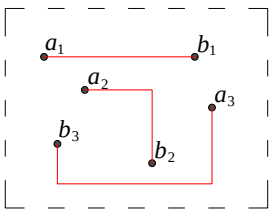


Figura 1.11: Emparejamiento ortogonal plano.

Como se observó anteriormente, una aproximación al diseño VLSI (Very Large Scale Integration) de circuitos integrados mediante trazados ortogonales consiste en considerar el circuito a diseñar como un conjunto de pares de vértices que deben unirse mediante aristas, con similares restricciones que en el problema general del trazado ortogonal de grafos, i.e., es conveniente minimizar el número de codos y el número de cruces.

En este sentido se plantea el problema EMPAREJAMIENTO ORTOGONAL SIMPLE PLANO, en el cual parejas de puntos en el plano deben unirse mediante aristas ortogonales con un único codo sin que éstas se

crucen. A una arista ortogonal con, a lo sumo, un codo llamaremos *arista ortogonal simple*.

Para cada par de puntos (en el plano) existen dos aristas ortogonales simple, excepto cuando $x_a = x_b$ o $y_a = y_b$, caso en el que existe una única arista ortogonal simple, en este caso sin codos.

Se denomina problema de *emparejamiento ortogonal simple plano* (EOSP) al siguiente problema de decisión:

EMPAREJAMIENTO ORTOGONAL SIMPLE PLANO (EOSP)
Entrada: Un conjunto de pares de puntos en el plano, $W = \{w_1, w_2, \ldots, w_n\}$.
Pregunta: ¿Existe un emparejamiento ortogonal simple para W? (i.e., ¿Existe una elección de aristas ortogonales simples, una por cada par, w_i, sin que dos aristas se crucen?)

Este problema, denominado originalmente en inglés *Single Bend Wiring (*SBW*)*, fue estudiado y resuelto por Raghavan *et al* [RCS86]. Estos autores presentaron un algoritmo que permite obtener la respuesta al problema en tiempo polinomial, $O(n^2)$. Si no existen pares de puntos con la misma abscisa o ordenada, para cada par de puntos existen exactamente dos aristas ortogonales simple. Codificándolas mediante una variable lógica para cada par de puntos, las entradas del problema EOSP se reducen a entradas del problema 2SAT en tiempo cuadrático. Dado que 2SAT es resoluble en tiempo lineal respecto al número de cláusulas de la fórmula proposicional [val00], la complejidad cuadrática resulta de la reducción del problema EOSP al problema 2SAT.

Adaptando el algoritmo desarrollado por Imai y Asano para resolver un problema de intersección de rectángulos [IA86] es posible resolver el problema EOSP en tiempo $O(nlog(n))$.

Para una entrada del problema EOSP con respuesta negativa, determinar el máximo subconjunto de pares de puntos para los cuales existe un conjunto de aristas ortogonales simple sin intersecciones, es un problema de optimización NP-duro. Es también NP-duro el problema de determinar el número mínimo de capas *(layers)* necesarias para que todos los pares de puntos puedan ser unidos por aristas ortogonales sin que dos aristas en una misma capa se crucen [RCS86].

El problema EOSP y otros similares tienen una clara aplicación al diseño de circuitos. La necesidad de evitar cruces entre las líneas de un circuito obliga frecuentemente a la existencia de *asas*, que se suelen realizar atravesando la placa lógica (v. Figura 2). Considerando que esta solución no tiene representación en el plano, Garrido, Marquéz, Morgana y Portillo [GMM02], estudiaron la generalización natural del problema de emparejamiento simple plano a otras superficies.

Para la representación ortogonal de grafos en superficies, en la definición anterior, en lugar de un sistema de ejes precisamos considerar un sistema ortogonal de líneas coordenadas. Como por ejemplo el sistema de referencia formado por el conjunto de los meridianos y los paralelos de una esfera.

Entonces una representación ortogonal de un grafo $G = (V, A)$ en una superficie S sobre la cual esté definido un sistema ortogonal de líneas coordenadas es un dibujo en el cual los vértices están representados por puntos y cada arista por una secuencia de segmentos consecutivos. Cada segmento corresponde a un arco de una línea coordenada. Con esta generalización todas las definiciones anteriores se mantienen con las debidas adaptaciones.

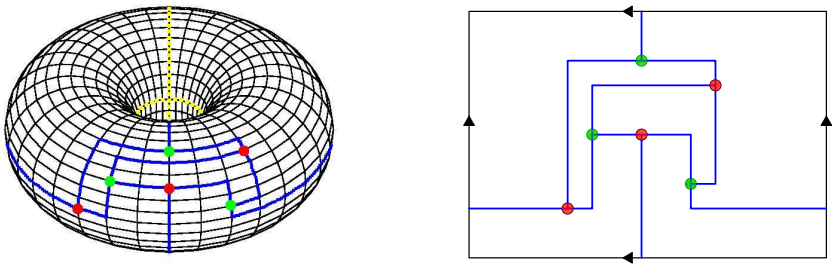


Figura 1.12: Trazado ortogonal del grafo bipartito completo $K_{3,3}$ en el toro, con aristas ortogonales simple (con un codo) y su representación ortogonal plana.

La Figura 1.12 presenta un trazado ortogonal del grafo bipartito $K_{3,3}$ en el toro. Cada arista ortogonal simple está constituida por un arco de meridiano y un arco de paralelo. Obsérvese que en la representación plana del toro los meridianos se representan mediante segmentos verticales (paralelos al eje de las y's) y los paralelos por segmentos horizontales (paralelos al eje de las x's).

En este sentido se plantea el problema EMPAREJAMIENTO ORTOGONAL SIMPLE sobre una superficie no plana, en la cual parejas de puntos deben unirse mediante aristas ortogonales con un único codo sin que éstas se crucen.

Problema que se formaliza a continuación:

EMPAREJAMIENTO ORTOGONAL SIMPLE en superficies (EOS) [GMM02]
Entrada: Una superficie S (de género g) y un conjunto de pares de puntos W en la superficie.
Pregunta: ¿Existe un emparejamiento ortogonal simple para W en S?

Imponiendo, como condición adicional, que para cada par de puntos sean admisibles a lo más r aristas ortogonales simples (previamente identificadas), obtenemos una colección de subproblemas que denominaremos r-EMPAREJAMIENTO ORTOGONAL SIMPLE (rEOS).

Cuando $r \leq 2$, el problema puede ser resuelto en tiempo polinomial. En cambio, para $r \geq 3$ se tiene que, al igual que el problema general EOS, los problemas rEOS son NP-completos [GMM02].

Los resultados citados en los párrafos anteriores fueron establecidos todos recurriendo al problema de la satisfacibilidad. Por ejemplo, cada una de las instancias del problema 2EOS puede ser reducida a una instancia del problema 2SAT en tiempo cuadrático respecto al número de pares de puntos. Entonces, el problema 2EOS puede ser resuelto en tiempo polinomial.

Las múltiples variantes del problema EOS considerando superficies particulares permanecían abiertas. Especialmente los problemas de emparejamiento ortogonal simple en el cilindro y en el toro, a los que nos referiremos más pormenorizadamente en las siguientes páginas.

§ 1.5.3. Emparejamientos ortogonales en el cilindro y en el toro.— Junto con el plano, la esfera, el cilindro y el toro son las superficies matemáticas más simples. En el estudio que sigue consideraremos las dos últimas: el cilindro y el toro. Una de las razones para considerar los problemas de emparejamiento en cada una de estas superficies proviene del hecho de que el problema EOSP tiene respuesta negativa en múltiples situaciones, incluso en entradas con apenas dos pares de puntos. Otra de las razones es el hecho de ser el problema EOS NP-completo.

La Figura 1.13 ilustra algunos ejemplos de entradas resolubles en una superficie pero no en otras. En el centro de la figura, el emparejamiento ortogonal simple es posible en el cilindro pero no en el plano y a la derecha, vemos un conjunto de pares de puntos cuyo emparejamiento ortogonal simple es posible en el toro pero no en el cilindro.

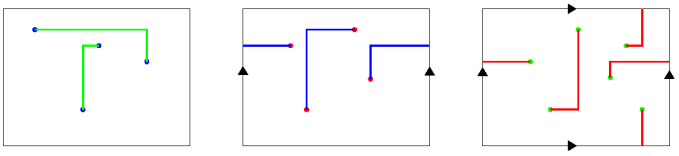


Figura 1.13: Izquierda: emparejamiento posible en el plano. Centro: emparejamiento posible en el cilindro pero no en el plano. Derecha: emparejamiento posible en el toro pero no en el cilindro.

En este sentido, el problema cilíndrico constituye una generalización del problema plano y el problema tórico, una generalización del problema cilíndrico.

EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL CILINDRO (EOSC) [por02]
Entrada: Un conjunto de pares de puntos sobre el cilindro.
Pregunta: ¿Existe un emparejamiento ortogonal simple para este conjunto?

EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL TORO (EOST) [por02]
Entrada: Un conjunto de distintos pares de puntos sobre el toro.
Pregunta: ¿Existe un emparejamiento ortogonal simple para este conjunto?

Tanto en una superficie genérica como en el toro, para cada par de puntos existen ocho posibles aristas ortogonales para unirlos. Portillo [por02] codificó las aristas utilizando tres variables lógicas, (p, m, h). En la representación plana de la superficie, tomando en cada par el punto de mayor ordenada, $h = 1$ si el segmento que incide en el punto es horizontal (arco de paralelo), $m = 1$ si la arista ortogonal corta el meridiano exterior de la representación plana y $p = 1$ si la arista corta el paralelo exterior. En la Figura 1.14 están representados las ocho aristas posibles y sus correspondientes codificaciones.

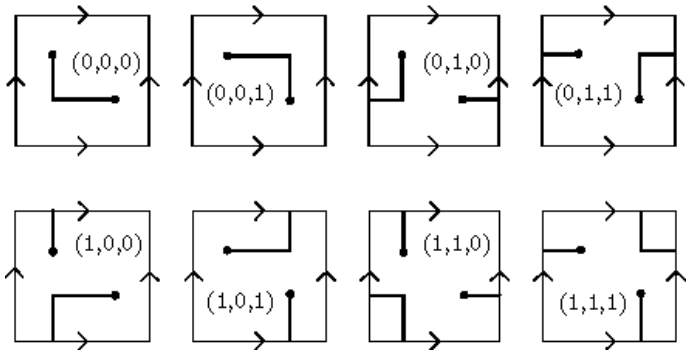


Figura 1.14: Aristas ortogonales simples uniendo un par de puntos en el toro.

En el caso particular de un par de puntos en el cilindro, existen cuatro aristas ortogonales posibles que corresponden a las dibujadas en la primera línea de la Figura 1.14, $p = 0$. En este caso serán suficientes dos variables lógicas (m, h). En el caso del plano, cada par de puntos

admite dos aristas ortogonales, $(0,0,0)$ y $(0,0,1)$, por lo que es suficiente una única variable lógica (h).

En el estudio de los problemas de emparejamiento ortogonal simple en el cilindro y en el toro asumiremos, sin perdida de generalidad, que los (dos) puntos de cada par están en meridianos y paralelos distintos. Por lo tanto, en la representación ortogonal estándar de estas dos superficies un par de puntos nunca están ambos en la misma recta horizontal o vertical. La presente simplificación tiene como único objetivo facilitar la exposición que sigue.

Con el objetivo de reducir cada instancia del problema EOSC al problema de satisfacibilidad booleana, Portillo [por02] estudió las 72 distintas posiciones posibles para cada dos pares de puntos y identificó 12 fórmulas proposicionales elementales, mostradas en el Cuadro 1.4.

Cuadro 1.4: Fórmulas proposicionales elementales en el problema EOSC (i<j).

	$\mathcal{F}_{i,j}$
I	$\{\}$
II	$\{[m_j]\}$
III	$\{[h_i, \overline{m_j}], [\overline{h_i}, m_j]\}$
IV	$\{[h_i, m_j], [\overline{h_i}, \overline{m_j}]\}$
V	$\{[\overline{m_j}]\}$
VI	$\{[h_i, \overline{m_i}], [\overline{h_j}, \overline{m_j}]\}$
VII	$\{[h_i, \overline{m_i}], [\overline{h_j}, m_j]\}$
VIII	$\{[h_i, m_i], [\overline{h_j}, \overline{m_j}]\}$
IX	$\{[h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j], [h_i, \overline{m_j}, \overline{h_j}], [\overline{h_i}, m_j, \overline{h_j}]\}$
X	$\{[h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j], [h_i, m_j, \overline{h_j}], [\overline{h_i}, \overline{m_j}, \overline{h_j}]\}$
XI	$\{[h_i, m_i, h_j], [h_i, \overline{m_i}, \overline{h_j}], [h_i, \overline{m_j}, \overline{h_j}], [\overline{h_i}, m_j, \overline{h_j}]\}$
XII	$\{[h_i, m_i, h_j], [h_i, \overline{m_i}, \overline{h_j}], [h_i, m_j, \overline{h_j}], [\overline{h_i}, \overline{m_j}, \overline{h_j}]\}$

Considerando el ejemplo de la Figura 1.15, las aristas ortogonales que no se cruzan corresponden a las soluciones de la siguiente ecuación lógica:

$$(h_i \wedge \overline{m_i} \wedge h_j \wedge \overline{m_j}) \vee (\overline{h_i} \wedge \overline{m_i} \wedge h_j) \vee (h_i \wedge m_i \wedge h_j \wedge \overline{m_j}) \vee \\ \vee (\overline{h_i} \wedge m_i \wedge \overline{h_j}) \vee (\overline{h_i} \wedge m_i \wedge h_j \wedge m_j) = 0.$$

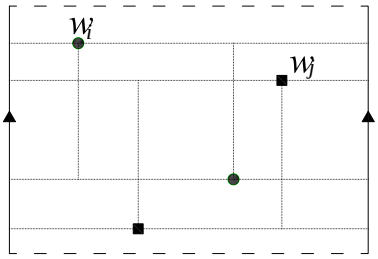


Figura 1.15: Instancia del problema EOSC con 2 pares de puntos.

Negando ambos miembros y utilizando las leyes de De Morgan se obtiene la ecuación,

$$(\overline{h_i} \vee m_i \vee \overline{h_j} \vee m_j) \wedge (h_i \vee m_i \vee \overline{h_j}) \wedge (\overline{h_i} \vee \overline{m_i} \vee \overline{h_j} \vee m_j) \wedge \\ (h_i \vee \overline{m_i} \vee h_j) \wedge (h_i \vee \overline{m_i} \vee \overline{h_j} \vee \overline{m_j}) = 1,$$

equivalente a

$$(\overline{h_i} \vee \overline{h_j} \vee m_j) \wedge (h_i \vee m_i \vee \overline{h_j}) \wedge (h_i \vee \overline{m_i} \vee h_j) \wedge (h_i \vee \overline{h_j} \vee \overline{m_j}) = 1.$$

Por lo que, en forma normal conjuntiva, se tiene la siguiente fórmula proposicional elemental (v. Ecuación IX, Cuadro 1.4),

$$\mathcal{F}_{i,j} = \{[\overline{h_i}, \overline{h_j}, m_j], [h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j], [h_i, \overline{h_j}, \overline{m_j}]\}.$$

Al efecto de la clasificación de las ecuaciones en el Cuadro 1.4, las entradas del problema se consideran ordenadas a partir de su representación ortogonal estándar. Específicamente, $w_i < w_j$ ($i < j$) si la ordenada del punto superior del par w_i es mayor que la ordenada del punto superior del par w_j.

De este modo, la fórmula proposicional en forma normal conjuntiva $\mathcal{F}$, respecto a la instancia del problema EOSC, se obtiene haciendo la conjunción de todas las fórmulas elementales obtenidas a partir de los pares de puntos analizados dos a dos. Por lo tanto, es posible reducir cualquiera de las instancias del problema EOSC al problema de satisfacibilidad en tiempo cuadrático respecto al número de pares de puntos de la entrada W, $O(|W|^2)$.

Complejidad que no se incrementa aunque sea necesario efectuar una ordenación previa de la entrada. Cuando el conjunto de entrada W no esté ordenado según este criterio, en una operación de preprocesamiento, se puede ordenar sus puntos en tiempo $O(nlog(n))$.

Siguiendo la misma metodología para el problema EOST, a cada par de puntos de una entrada le corresponde una de las 21 fórmulas proposicionales que se presentan en el Cuadro 1.5.

Cuadro 1.5: Fórmulas proposicionales elementales en el problema EOST ($i < j$).

	$\mathcal{F}_{i,j}$
1	$\{[p_i, m_j], [\overline{m_i}, \overline{p_j}]\}$
2	$\{[p_i, \overline{m_j}], [m_i, \overline{p_j}]\}$
3	$\{[p_i, \overline{m_j}], [\overline{m_i}, \overline{p_j}]\}$
4	$\{[\overline{p_i}, m_j], [\overline{m_i}, \overline{p_j}]\}$
5	$\{[\overline{p_i}, \overline{m_j}], [m_i, \overline{p_j}]\}$
6	$\{[\overline{p_i}, \overline{m_j}], [\overline{m_i}, \overline{p_j}]\}$
7	$\{[h_i, m_i, p_j], [\overline{h_i}, m_i, \overline{p_j}], [p_i, \overline{h_j}, \overline{m_j}], [\overline{p_i}, h_j, \overline{m_j}]\}$
8	$\{[h_i, \overline{m_i}, p_j], [\overline{h_i}, \overline{m_i}, \overline{p_j}], [p_i, \overline{h_j}, \overline{m_j}], [\overline{p_i}, h_j, \overline{m_j}]\}$
9	$[h_i, \overline{m_i}, p_j], [\overline{h_i}, \overline{m_i}, \overline{p_j}], [p_i, \overline{h_j}, m_j], [\overline{p_i}, h_j, m_j]\}$
10	$\{[h_i, p_i, m_j], [\overline{h_i}, p_i, \overline{m_j}], [m_i, h_j, \overline{p_j}], [\overline{m_i}, \overline{h_j}, \overline{p_j}]\}$
11	$[h_i, p_i, m_j], [\overline{h_i}, p_i, \overline{m_j}], [m_i, \overline{h_j}, \overline{p_j}], [\overline{m_i}, h_j, \overline{p_j}]\}$
12	$\{[h_i, p_i, \overline{m_j}], [\overline{h_i}, p_i, m_j], [m_i, h_j, \overline{p_j}], [\overline{m_i}, \overline{h_j}, \overline{p_j}]\}$
13	$\{[h_i, p_i, \overline{m_j}], [\overline{h_i}, p_i, m_j], [m_i, \overline{h_j}, \overline{p_j}], [\overline{m_i}, h_j, \overline{p_j}]\}$
14	$\{[h_i, \overline{p_i}, m_j], [\overline{h_i}, \overline{p_i}, \overline{m_j}], [m_i, h_j, \overline{p_j}], [\overline{m_i}, \overline{h_j}, \overline{p_j}]\}$
15	$\{[h_i, \overline{p_i}, m_j], [\overline{h_i}, \overline{p_i}, \overline{m_j}], [m_i, \overline{h_j}, \overline{p_j}], [\overline{m_i}, h_j, \overline{p_j}]\}$
16	$\{[h_i, \overline{p_i}, \overline{m_j}], [\overline{h_i}, \overline{p_i}, m_j], [m_i, h_j, \overline{p_j}], [\overline{m_i}, \overline{h_j}, \overline{p_j}]\}$
17	$\{[h_i, \overline{p_i}, \overline{m_j}], [\overline{h_i}, \overline{p_i}, m_j], [m_i, \overline{h_j}, \overline{p_j}], [\overline{m_i}, h_j, \overline{p_j}]\}$
18	$\{[h_i, h_j, \overline{m_i}, p_j], [h_i, \overline{h_j}, m_i, p_j], [\overline{h_i}, h_j, \overline{m_i}, \overline{p_j}], [\overline{h_i}, \overline{h_j}, m_i, \overline{p_j}],$ $[h_i, h_j, \overline{m_j}, \overline{p_i}], [h_i, \overline{h_j}, \overline{m_j}, p_i], [\overline{h_i}, \overline{h_j}, m_j, p_i], [\overline{h_i}, h_j, m_j, \overline{p_i}]\}$
19	$\{[h_i, h_j, \overline{m_i}, p_j], [h_i, \overline{h_j}, m_i, p_j], [\overline{h_i}, h_j, \overline{m_i}, \overline{p_j}], [\overline{h_i}, \overline{h_j}, m_i, \overline{p_j}],$ $[h_i, h_j, m_j, \overline{p_i}], [h_i, \overline{h_j}, m_j, p_i], [\overline{h_i}, h_j, \overline{m_j}, \overline{p_i}], [\overline{h_i}, \overline{h_j}, \overline{m_j}, p_i]\}$
20	$\{[h_i, h_j, m_i, p_j], [h_i, \overline{h_j}, \overline{m_i}, p_j], [\overline{h_i}, h_j, m_i, \overline{p_j}], [\overline{h_i}, \overline{h_j}, \overline{m_i}, \overline{p_j}],$ $[h_i, h_j, \overline{m_j}, \overline{p_i}], [h_i, \overline{h_j}, \overline{m_j}, p_i], [\overline{h_i}, h_j, m_j, \overline{p_i}], [\overline{h_i}, \overline{h_j}, m_j, p_i]\}$
21	$\{[h_i, h_j, m_i, p_j], [h_i, \overline{h_j}, \overline{m_i}, p_j], [\overline{h_i}, h_j, m_i, \overline{p_j}], [\overline{h_i}, \overline{h_j}, \overline{m_i}, \overline{p_j}],$ $[h_i, h_j, m_j, \overline{p_i}], [h_i, \overline{h_j}, m_j, p_i], [\overline{h_i}, h_j, \overline{m_j}, \overline{p_i}], [\overline{h_i}, \overline{h_j}, \overline{m_j}, p_i]\}$

Para posteriores referencias las fórmulas del problema EOST se refieren con caracteres arábigos y las del problema EOSC con caracteres romanos.

Denotaremos EOSC-SAT y EOST-SAT a las clases de satisfacibilidad constituidas por las fórmulas proposicionales que correspondan a alguna entrada del problema de emparejamiento correspondiente. Atendiendo a las expresiones de las fórmulas elementales correspondientes, EOSC-SAT es una subclase propia de 3SAT y EOST-SAT es una subclase propia de 4SAT.

Observamos anteriormente algunos aspectos de los problemas de emparejamiento ortogonal simple en el toro y en el cilindro. De un modo análogo se podría definir el mismo problema en la esfera. Sin embargo esta superficie presenta aspectos peculiares. Desde el punto de vista topológico, esta superficie es equivalente al plano, i.e., cualquier grafo plano admite una inmersión en la esfera y viceversa. Por otro lado, para conectar un par de puntos sobre la esfera, y a semejanza del toro, existen hasta ocho trazados ortogonales simples, constituidos por un arco de meridiano y/o un arco de paralelo. Aún más, para una entrada con varios pares de puntos, si un par está unido por un trazado que pasa por los polos polos, para cada uno de los demás pares quedan (como en el cilindro) cuatro posibilidades de trazado.

Otro aspecto que nos lleva a no considerar la esfera en el estudio de emparejamientos ortogonales en superficies particulares, es el hecho de la representación ortogonal plana: en esta superficie los meridianos y paralelos no se corresponden con segmentos de recta horizontales y verticales, contrariamente a lo que ocurre en el cilindro y en el toro. Sin embargo, la esfera puede y debe ser objeto de un estudio posterior, complementario al presentado en este trabajo.

1.6. Problemas de etiquetado

La producción automática de cartografía constituye una de las importantes funcionalidades de un Sistema de Información Geográfica (SIG). En cartografía, la utilización de texto junto a entidades gráficas puntuales (p.e. indicando el nombre de ciudades), lineales (con la identificación de una autopista) o de área (con el nombre de un lago), constituye uno de los elementos principales de un mapa. Llamamos *etiquetar un mapa* a la aplicación de elementos de texto sobre un mapa.

La automatización de esta operación ha sido considerada por la Computational Geometry Task Force [cha96, CET99] (ACM) como una

de las áreas más importantes de investigación en el dominio de la Geometría Computacional.

Los primeros pasos en este sentido aparecen en 1934, en Francia, en un trabajo elaborado por el *Service Géographique de l'Armée* [sgl34]. Posteriormente, Imhof [imh75] identifica un conjunto de criterios que se deben observar en la colocación de las etiquetas en un mapa. Estos criterios son fundamentalmente estéticos: el tamaño de las etiquetas debe permitir la legibilidad del texto, la localización debe ser tal que evite ambigüedades, etc. Para ilustrar estos objetivos, este autor presenta 100 ejemplos de buenas y malas decisiones de etiquetado.

En este sentido, considerando los diferentes criterios para una adecuada localización de las etiquetas de un conjunto de entidades, las operaciones de etiquetado de un mapa se pueden subdividir en tres clases según el elemento gráfico representado:

1. etiquetar áreas, tales como países u océanos;

2. etiquetar líneas (segmentos) como, por ejemplo, ríos o vías de comunicación

3. etiquetar puntos, tales como ciudades y puntos con cotas.

Independientemente de la naturaleza de cada entidad a etiquetar, las etiquetas no deben sobreponerse unas a otras. Este es, con toda seguridad, el principal problema a resolver para la colocación automática de etiquetas [CMS95]. Un algoritmo eficiente debe efectuar la colocación de las etiquetas para cada una de las tres clases sin superposición.

Etiquetar un mapa requiere un cartógrafo para considerar los muchos criterios en conflicto, tales como localización, orientación, forma, tamaño y características del texto, para cada una de las etiquetas a colocar. Manualmente, es una tarea fastidiosa estimándose que ocupa cerca de la mitad del tiempo necesario para la elaboración de un mapa.

Más aún, respecto al aspecto combinatorio entre las etiquetas, Marks y Shieber [MS91], señalan otros aspectos a considerar en la aplicación de etiquetas, específicamente:

- el nivel de superposición entre etiquetas y el modo como oculta o obscurece las entidades gráficas de un mapa;

- el grado de claridad en el sentido de cómo las etiquetas están asociadas a los elementos gráficos correspondientes;

- las preferencias, *a priori*, de entre las posiciones posibles y teóricamente aceptables, para la colocación de las etiquetas;

- el número de entidades gráficas por etiquetar.

Asumimos la legibilidad como prioridad sobre los aspectos estéticos, particularmente en documentos y mapas técnicos en los cuales todas las entidades tienen que ser etiquetadas.

Estas consideraciones conducen a la necesidad de encarar el problema de etiquetado de un modo más generalizado, que excede el propósito de esta nota introductoria. Una extensa y actualizada bibliografía sobre el problema de etiquetado puede encontrarse en [WS96]. Su consulta permite, además, evaluar el interés de la comunidad científica por este tema.

Debido a la intratabilidad del problema y con el objetivo de su descomposición en casos particulares más simples que permitiesen desarrollar su estudio sistemáticamente por trozos, se establece un criterio de clasificación basado en tres aspectos. Según la naturaleza de las entidades a etiquetar: puntos, líneas y áreas; según las características de las etiquetas a utilizar (forma, tamaño,área, etc) y según el posicionamiento de las etiquetas respecto a las entidades a etiquetar (fijas en un punto, en posiciones contiguas, etc).

Con frecuencia, los elementos a etiquetar son puntos y las etiquetas con la información, rectangulares con sus medidas previamente fijadas. En estos problemas de etiquetado existen dos distintos modelos. Por un lado tenemos el modelo introducido por Forman y Wagner [FW91], que llamamos de etiquetas fijas, en el cual cada etiqueta puede ser colocada en un determinado número de posiciones previamente determinadas. Alternativamente, las etiquetas pueden colocarse en posiciones variable contiguas al objeto etiquetable y se denominan etiquetas deslizantes. En estas condiciones se pueden plantear principalmente dos preguntas. De decisión, que consiste en determinar si las etiquetas (rectángulos) pueden ser colocadas sin que exista superposición, y de optimización, cuando el objetivo es posicionar las etiquetas en las mejores posiciones.

En el caso de las etiquetas en posiciones prefijadas, muchas veces se pretende que la etiqueta esté posicionada de modo que el punto coincida con uno de los cuatro vértices del rectángulo. Incluso con estas restricciones el problema sigue siendo computacionalmente intratable. Considerando un conjunto de puntos, y para cada punto una etiqueta de tamaño unitario, el problema de existencia de una solución de modo que un vértice de cada cuadrado coincida con el punto a etiquetar (y sus lados sean paralelos a los ejes coordenados) es un problema de decisión NP-completo [FW91].

Con frecuencia, se dan situaciones prácticas dónde los puntos a etiquetar se encuentran localizados sobre una recta. El mapa del metro de Lisboa, en la Figura 1.16 ilustra uno de estos casos. Los puntos a etiquetar se encuentran alineados y las etiquetas inscritas en rectángulos cuyos lados son horizontales y verticales.

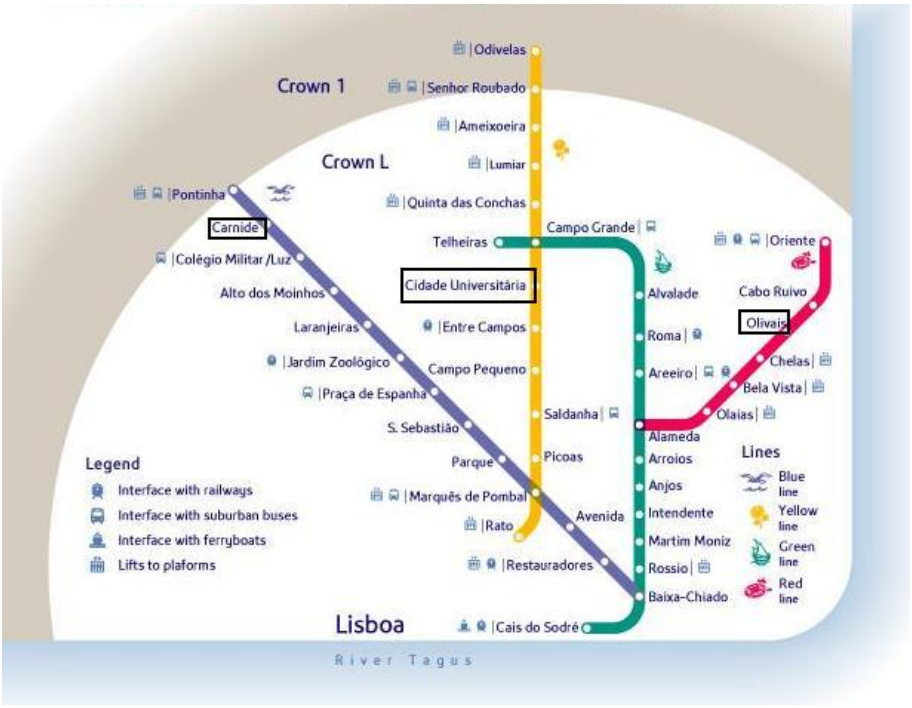


Figura 1.16: Mapa del metro de la ciudad de Lisboa.

El estudio de este problema específico de etiquetar puntos sobre una línea horizontal, vertical o oblicua, con etiquetas cuadradas o rectangulares, en posición fija o deslizantes, puede encontrarse en varios trabajos, p.e., Chen, Lee y Liao (2005) [CLL05], Chen [che03], Reyes (2002) [rey02], Garrido *et al.* (2001) [GIM01], Poon *et al.* (2001) [PSS01], y Shin y Yang (2001) [SY01].

P. Reyes, en su tesis de doctorado *Problemas de Etiquetado: Complejidad Computacional*, estudia diversos problemas de etiquetado de puntos alineados, en posición fija, que se refieren en el Cuadro 1.6. En el mismo trabajo demuestra que, para etiquetas en posición fija, cuando la recta de soporte de los puntos es horizontal o vertical el problema es resoluble en tiempo polinomial (v. problema H-4P, Cuadro 1.6).

Cuando la recta está en posición oblicua (relativamente a los ejes coordenados) la complejidad del problema depende del tipo de etiquetas. Si las etiquetas son cuadradas o rectangulares, todas del mismo tamaño, OC-4P y OR1-4P respectivamente, el problema es resoluble en tiempo lineal respecto al número de puntos a etiquetar, $O(n)$. El problema tiene también resolución en tiempo lineal, $O(n)$, si las etiquetas son rectangulares con altura unitaria, O1R-4P (v. Cuadro 1.6).

Cuadro 1.6: Problemas de decisión de etiquetado de puntos alineados con etiquetas en posición fija y su complejidad computacional.

Problema	Posición de los puntos	Tipo de etiquetas	Complejidad computacional
H–4P	línea horizontal	rectángulos arbitrarios	$O(n)$
OC–4P	línea oblicua	cuadrados iguales	$O(n)$
OR(1)–4P	línea oblicua	rectángulos iguales	$O(n)$
OR(m)–4P	línea oblicua	rectángulos de m tamaños diferentes	$O(2^{cm}n)$
O1R–4P	línea oblicua	rectángulos de igual altura	$O(n)$

Considerando el problema de etiquetar puntos alineados con etiquetas rectangulares en cuatro posiciones posibles y suponiendo que existen $m(\leq n)$ rectángulos de tamaños distintos, OR(m)-4P, este autor propone un algoritmo incremental que permite la resolución de este problema en tiempo pseudopolinomial, $O(2^{cm}n)$, dónde c designa una constante. Por lo tanto, si los rectángulos fueran todos de diferentes tamaños, $(m = n)$, el algoritmo correría en tiempo exponencial.

En el caso que la recta de soporte tenga pendiente unitaria y las etiquetas sean cuadradas (con distintos tamaños), UC-4P, mostraremos más adelante que este problema de decisión, puede ser resuelto en tiempo polinomial. Para resolverlo consideraremos su reducción al problema de satisfacibilidad, más propiamente al problema de satisfacibilidad de la clase UC-4P-SAT, que constituye una clase de satisfacibilidad polinomial.

1.7. Clases de satisfacibilidad polinomiales (bien conocidas)

El problema de la satisfacibilidad SAT es un problema NP-completo, es más, fue el primer problema NP-completo conocido. Por tanto, a menos que P=NP, no existe ningún algoritmo polinomial que lo resuelva.

Sin embargo, mediante ciertas restricciones pueden obtenerse subclases de fórmulas proposicionales para las cuales son conocidos algoritmos deterministas polinomiales que resuelven el problema de la satis-

facibilidad de sus instancias. Estas subclases son designadas por *clases de satisfacibilidad polinomiales* o, abreviadamente, *clases polinomiales*. Por oposición, las otras clases de satisfacibilidad para cuyas instancias el problema de satisfacibilidad sea tan difícil como la resolución del problema SAT, serán llamadas clases de satisfacibilidad NP-completas o, abreviadamente, clases NP-completas. Las clases kSAT, con $k \geq 3$, constituyen ejemplos de clases de satisfacibilidad NP-completas.

Además de la clase 2SAT, cuyas fórmulas están constituidas por cláusulas con dos literales, existen otras clases polinomiales. En un trabajo relativamente reciente, Franco y Gelder (2003) [FG03] presentan un estudio sobre el comportamiento de fórmulas kSAT (k constante, independiente del número de cláusulas), generadas aleatoriamente, con todas las cláusulas conteniendo exactamente k literales, investigando la probabilidad de que las fórmulas generadas pertenezcan a alguna clase polinomial conocida. Estos autores sugieren que la aproximación al problema desarrollada en este estudio debe constituirse en una línea de orientación para el estudio de nuevas clases de satisfacibilidad polinomiales.

A este efecto consideran un conjunto de clases polinomiales, que designan como *bien conocidas*: Horn, 2SAT, CC-*balanced*, *renamable* Horn, *extended* Horn, SLUR y LinAut. Añadiremos a estas la clase *matched formulas*, presentada por los mismos autores en el citado artículo.

Una breve referencia a otras clases polinomiales de satisfacibilidad se hace al final de esta sección.

Una fórmula proposicional en forma normal conjuntiva se llama *Horn* si cada una de las cláusulas contiene, a lo máximo, un literal afirmado. Se trata de una clase ampliamente estudiada, en parte por su asociación con la programación lógica [IM82, DG84, scu90]. Las instancias de esta clase pueden ser resueltas en tiempo lineal utilizando el proceso de simplificación conocido por *resolución unitaria* [DP60]. Siempre que exista una cláusula unitaria, $C = [l]$, el literal de esa cláusula es declarado verdadero (atribuido con el valor 1), todas las cláusulas que lo contengan son eliminadas de la fórmula y el complementario del literal, $\bar{l}$, es eliminado de cada una de las cláusulas que lo contenga. La iteración de esta simplificación es conocida como *propagación unitaria* y el correspondiente algoritmo se designa como PROPUNIT, que será presentado más adelante.

Dada una fórmula proposicional $\mathcal{F}$, se llama matriz cláusula-variable a la matriz de m líneas y n columnas cuyas columnas se corresponden con las variables de $\mathcal{F}$ ($var(\mathcal{F})$) y las líneas con las cláusulas de la fórmula proposicional. Considerando una cláusula C_i, representada por la línea i de la matriz, la entrada M_{ij} tiene el valor $+1$ si la cláusula contiene el literal afirmado sobre la variable v_j y -1 se contiene el literal negado. Caso contrario, se tiene $M_{ij} = 0$.

Sea $\mathcal{F}$ una fórmula proposicional y M su correspondiente matriz cláusula-variable . Si la fórmula proposicional correspondiente a la matriz cláusula-variable que se obtiene de M multiplicando un subconjunto de columnas por -1 es una fórmula Horn, se dice entonces que $\mathcal{F}$ es una fórmula proposicional *renamable Horn* [asp80]. Estas fórmulas pueden ser reconocidas y resueltas en tiempo polinomial [val00].

Una fórmula proposicional se denomina *CC-balanced* si para cualquier submatriz de M (matriz cláusula-variable) con exactamente dos entradas no nulas por línea y por columna, la suma de las entradas es un múltiplo de 4 [tru82, CGK94]. En este caso el problema de la satisfacibilidad de la fórmula proposicional se reduce al problema de programación lineal $\{x : Mx \geq 1 - n(M), 0 \leq x \leq 1\}$, dónde $n(M)$ es el vector con m entradas corresponden al número de literales negados en la correspondiente cláusula. La satisfacibilidad de una fórmula *CC-balanced* puede ser determinada en tiempo polinomial recurriendo a algoritmos polinomiales de programación lineal [GPF97]. La matriz cláusula-variable de una fórmula *CC-balanced* es una *matriz equilibrada*. Considerando que este tipo de matriz puede ser reconocida en tiempo polinomial [zam05], el problema de reconocimiento de las fórmulas proposicionales *CC-balanced* puede ser resuelto en tiempo polinomial.

Una extensión de la clase Horn, designada por extended *Horn*, fue propuesta por Chandru y Hooker [CH91] en un trabajo de investigación sobre las condiciones que deben darse para que las técnicas de programación lineal puedan ser utilizadas en la resolución de instancias de SAT. Una caracterización alternativa, a partir de grafos orientados, puede encontrarse en [SAF95].

Chandru y Hooker demostraron que el algoritmo de propagación unitaria puede ser utilizado en la resolución de instancias de esta clase y presentaron un método para determinar la solución cuando no existan cláusulas unitarias. Sin embargo, esta técnica depende del reconocimiento previo de las instancias, lo que constituye una dificultad pues no es conocido ningún algoritmo polinomial que permita decidir si una fórmula proposicional dada pertenece o no a esta clase [SAF95].

Para resolver el problema de la satisfacibilidad de una fórmula proposicional de esta clase sin necesidad del reconocimiento previo, Schlipf *et al* [SAF95] desarrollaron el algoritmo SLUR. Constatándose que el mismo algoritmo permite resolver una amplia clase de fórmulas proposicionales, Franco y Gelder [FG03] definen una clase que designaron también por SLUR. La clase SLUR contiene algunas de las otras clases de satisfacibilidad bien conocidas. En particular, contiene a las clases Horn, *renamable* Horn, CC-*balanced* y *extended* Horn (v. Figura 1.17).

Consideremos M la matriz cláusula-variable de una fórmula propo-

sicional y el sistema de inecuaciones siguiente:

$$Mx \leq 2z1 - v(m), -1 \leq x \leq 1 \tag{1.1}$$

dónde $v(m)$ designa el vector de dimensión m cuyas entradas corresponden al número de literales en cada una de las cláusulas. Si la desigualdad es verdadera para $z = 1$, la fórmula se denomina q-Horn [BHS94, BCH94, scu90].

Considerando la desigualdad anterior, Boros *et al* establecieron el siguiente criterio que permite clasificar la *intratabilidad* de algunas clases de fórmulas proposicionales [BHS94, BCH94].

Teorema 1.4. *Para cualquier constante $c > 0$, la clase de fórmulas proposicionales que satisface el sistema de inecuaciones (1.1) con $z = 1 + \frac{c \cdot log(n)}{n}$ puede ser resuelta en tiempo polinomial.*

Para cualquier constante $\beta < 1$, la clase de fórmulas que satisface el sistema (1.1) con $z = 1 + \frac{1}{n^\beta}$ es NP-completa.

Este resultado llevó a Gu, Purdom, Franco y Wah, en 1997, a conjeturar la existencia de una *superclase* de satisfacibilidad polinomial [GPF97], conjetura reformulada por Granco y Gelder en 2003 [FG03].

La clase de satisfacibilidad LinAut fue presentada por Kullmann [kul00], basándose en el concepto de *asignaciones de autarquías* que, en cierto sentido, se relaciona con la existencia de *particiones* en una fórmula proposicional. Dada una fórmula proposicional $\mathcal{F}$, las *autarquías* son atribuciones de verdad parciales satisfaciendo alguna fórmula proposicional $\mathcal{F}' \subset \mathcal{F}$ sin que las cláusulas de $\mathcal{F} - \mathcal{F}'$ sean afectadas por la atribución. En general, determinar una *asignación de autarquía* (si existe), es un problema NP-completo [kul00].

Una primera versión de la descomposición de una fórmula proposicional cuando el núcleo de la matriz cláusula-variable es no nulo puede ser encontrado en el trabajo de Maaren y Warners [WM00a]. Kullmann [kul00] define el concepto de *autarquía lineal*, que generaliza el anterior concepto de descomposición y muestra que utilizando algoritmos de programación lineal, es posible determinar una *asignación de autarquía* en tiempo polinomial.

Considerando una fórmula proposicional $\mathcal{F}$, la correspondiente matriz cláusula-variable M admite una *autarquía lineal* $\mathbf{x} \in \mathbb{Q}^n$ si

$$\begin{cases} M\mathbf{x} \geq 0 \\ \mathbf{x} \neq 0 \end{cases} \tag{1.2}$$

Si existe una *autarquía lineal*, $\mathbf{x} \in \mathbb{Q}^n$ también existe una *asignación de autarquía*, τ, tal que el literal afirmado $v_j \in \tau$ si $sgn(x_j) = 1$ y $\overline{v_j} \in \tau$ si $sgn(x_j) = -1$.

Maaren [maa00] demostró que LinAut constituye una extensión de la clase q-Horn y que es incomparable con la clase SLUR.

La Figura 1.17 presenta las relaciones de inclusión entre las clases de satisfacibilidad polinomiales bien conocidas, representando la inclusión por el símbolo $\rightarrow$. La clase SLUR contiene propiamente a las subclases Horn, *renamable* Horn, *extended* Horn y CC-*balanced* y la clase LinAut contiene a las clases Horn, q-Horn, 2SAT y *matched formulas*. Es decir, SLUR y LinAut contienen entre ambas a todas las demás clases de satisfacibilidad polinomiales bien conocidas.

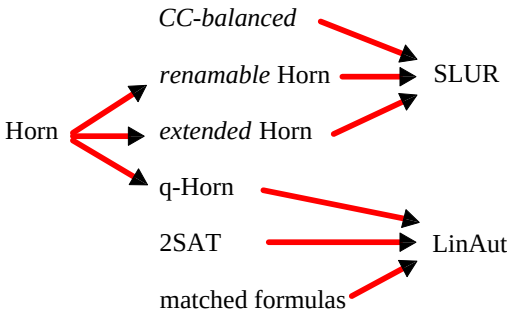


Figura 1.17: Clases polinomiales bien conocidas y sus relaciones de inclusión.

En el presente trabajo presentaremos una nueva clase de satisfacibilidad polinomial que contiene propiamente a las dos clases SLUR y LinAut y, por consiguiente, a todas las demás clases de satisfacibilidad polinomiales bien conocidas. Queda abierto el problema de si esta nueva clase, que designaremos por PURL, en algún sentido, corresponde a la superclase de satisfacibilidad polinomial conjeturada por Gu, Purdom, Franco y Wah [GPF97] y nuevamente sugerida por Franco y Gelder [FG03].

Más allá de las clases de satisfacibilidad polinomiales bien conocidas, podemos encontrar algunas otras en la literatura especializada, la mayor parte de las veces asociadas a problemas de decisión específicos.

Si bien es cierto que las instancias de algunos de los problemas polinomiales a los que nos hemos referido anteriormente, concretamente EOSP y 2EOS pueden ser reducidas a instancias de la clase de satisfacibilidad 2SAT, una de las clases de satisfacibilidad bien conocidas, no es fácil probar que esto sucede en otros casos, para otros problemas y otras clases de satisfacibilidad polinomiales.

Veremos por ejemplo en este trabajo, que UC-4P-SAT constituye una

subclase de 4SAT y EOSP una subclase de 3SAT, cuyas instancias no pertenecen a ninguna de las clases de satisfacibilidad polinomiales bien conocidas, aunque ambos son problemas resolubles en tiempo polinomial. Esta es una de las razones que nos lleva a formalizar la nueva clase de satisfacibilidad polinomial PURL.

El SUDOKU es un popular rompecabezas que se volvió internacionalmente conocido en 2005. Está constituido por un conjunto de 9×9 células, formando 9 bloques de 3×3 células en cada bloc. Algunas de las células contienen números entre 1 y 9 en tanto que otras se encuentran en blanco. Resolver un rompecabezas SUDOKU consiste en completar las células en blanco con números entre 1 y 9 de modo que cada línea, cada columna y cada bloque (de 3×3 células) contenga todos los números de 1 a 9. Además de esta caracterización, los rompecabezas publicados en la literatura de entretenimiento poseen dos propiedades adicionales: (1) cada rompecabezas tiene solución única y (2) su resolución puede ser conseguida por inferencia directa sin necesidad de proceder por intento y error para excluir posibles secuencias.

Lynce y Ouaknine [LO06] muestran dos formas para reducir las instancias del problema SUDOKU en las condiciones anteriores a instancias SAT, designadas por *codificación mínima* y *codificación ampliada*. La primera está formada por un mínimo de cláusulas que permite caracterizar cada rompecabezas y la segunda codificación incluye condiciones redundantes respecto a la codificación mínima. Además de las cláusulas unitarias, cada fórmula contiene 8829 cláusulas en la codificación mínima y 11988, en la ampliada.

Considerando un conjunto de 24260 rompecabezas SUDOKU con elevado grado de dificultad y utilizando exclusivamente esquemas de inferencia, —propagación unitaria (*PropUnit*) y propagación unitaria combinada con *failed literal rule* (*PropUnit+FLR*)—, Lynce y Ouaknine comprobaron que estos dos esquemas son más eficientes para la codificación ampliada. *PropUnit* no resolvió ninguna fórmula proposicional con la codificación mínima y resolvió cerca del $1\,\%$ de las fórmulas obtenidas con la codificación ampliada. Con el segundo esquema, *PropUnit+FLR*, cerca de $47\,\%$ de las codificaciones mínimas y $100\,\%$ de las codificaciones ampliadas pudieron ser resueltos.

Estos resultados ilustran que la estrategia de reducción a instancias SAT puede influir en la resolución de los problemas mediante este tipo de técnica. Ambas codificaciones fueron probadas con el algoritmo ELIMINALITERALES($\mathcal{F}$) y ELIMINALITERALES($\mathcal{F}, 2$) y los resultados se muestran al final del próximo capítulo, en la Sección 2.5.

Para concluir observemos que el problema SUDOKU puede ser fácilmente generalizado, considerando rompecabezas con $n^2\times n^2$ células y que este problema es NP-completo [YS03]. Sin embargo, los rompecabezas del problema SUDOKU generalizado que verifiquen las propiedades

(1) y (2) antes referidas son resolubles en tiempo polinomial [LO06].

Se observó anteriormente que se conocen varias clases de satisfacibilidad resolubles en tiempo polinomial, incluyendo las clases Horn y 2SAT. En el caso particular de estas dos, la resolución de las respectivas instancias puede ser efectuada en tiempo lineal respecto al tamaño de la instancia utilizando el algoritmo de propagación unitaria y considerando algunas propiedades específicas. Para las instancias Horn, cada una de las cláusulas con dos o más literales contiene un literal negado. Por lo tanto, la atribución de verdad conteniendo todos los literales negados constituye un modelo para cualquier instancia Horn que no contenga cláusulas unitarias. Para una instancia 2SAT, la propagación unitaria de una cualquier instancia $\mathcal{F}$ con una atribución de verdad parcial, τ, $(\mathcal{F}^U, \tau^U) = PropUnit(\mathcal{F} \cup \mathcal{U}(\tau))$, sigue siendo una instancia 2SAT. Se $\mathcal{F}^U$ no contiene la cláusula nula, esta fórmula y la original son equisatisfacibles.

Es conocida la existencia de fórmulas proposicionales que, aunque no pertenecen a ninguna de las dos clases anteriores (Horn y 2SAT), tomando una atribución de verdad τ, $\mathcal{F}^U$ si es una instancia de Horn o 2SAT y por lo tanto resoluble en tiempo polinomial. Tomando esta idea como punto de partida, Dalal y Etherington [DE92] definen una jerarquía de clases de satisfacibilidad $\Omega_0 \subset \Omega_1 \subset \cdots \subset \Omega_r \subset \ldots$ a partir de las clases polinomiales 2SAT y Horn, que verifica las siguientes propiedades: Ω_0 contiene a las clases polinomiales 2SAT y Horn y, para cada clase Ω_r, las entradas pueden ser resueltas en tiempo $O(mn^r)$, dónde m designa el número de cláusulas y n el número de variables de $\mathcal{F}$.

Otras referencias sobre jerarquías de clases de satisfacibilidad pueden ser encontradas en los trabajos de Pretolani (1996) [pre96], y de Gallo y Scutella (1988) [GS88].

§ 1.7.1. Otras clases de satisfacibilidad polinomiales.— Además de las clases de satisfacibilidad polinomiales bien conocidas mostradas en la Figura 1.17: Horn, *renamable* Horn, *extended* Horn, q-Horn, 2SAT, CC-*balanced*, *matched*, SLUR y LinAut, hay otras referenciadas en la bibliografía. De entre estas podemos destacar las clases QUAD [dal96], CoE [WM00b] y la clase que Benhamou [ben04] designó como $\mathcal{S}$.

La clase de satisfacibilidad polinomial QUAD, está formada por las CNF-fórmulas cuya satisfacibilidad puede ser resuelta en tiempo $O(n^2k)$, dónde n corresponde al tamaño de la entrada y k el mayor número de literales en una cláusula. La idea base es substituir repetidamente cláusulas en la fórmula por subcláusulas que pueden ser inferidas. Una fórmula pertenece a la clase QUAD si y sólo si durante este proceso se determina su satisfacibilidad o su no satisfacibilidad. Para inferir subcláusulas y determinar la satisfacibilidad se utiliza un algoritmo li-

neal llamado ROOT, que implementa propagación unitaria. Dado que para una cláusula pueden ser inferidas varias subcláusulas, este autor adopta una ordenación de las cláusulas para determinar cuál de entre las distintas cláusulas debe substituir a la cláusula original. QUAD depende, por tanto, de la ordenación adoptada. Las instancias de QUAD pueden ser reconocidas en tiempo polinomial, $O(n^2k)$.

Aunque existen algunos parecidos entre QUAD y PURL, esta última fue desarrollada independientemente de aquella (QUAD) y presenta varias y notables diferencias. PURL contiene a todas las clases de satisfacibilidad polinomiales bien conocidas en cuanto que QUAD no contiene ni a q-Horn ni a *renamable* Horn como subclases [dal96]. Recordemos que *renamable* Horn es subclase de SLUR y q-Horn de LinAut, por lo que QUAD no contiene como subclase a ninguna de estas últimas, SLUR y LinAut.

CoE, cuyas instancias son conjunciones de equivalencias, es también una clase de satisfacibilidad resoluble en tiempo polinomial [sch78]. Las fórmulas CoE se conocen también como fórmulas XOR (OR exclusivo). A pesar de ser resolubles en tiempo polinomial, para una instancia de esta clase en forma normal conjuntiva, un algoritmo basado en resolución corre en tiempo exponencial [tse70]. En 2000, Warners y Maareen [WM00b] mostraron que una CNF-fórmula doblemente equilibrada (*doubly balanced formula*) es equivalente a una fórmula CoE que se puede resolver en tiempo polinomial. En el mismo trabajo, sus autores resuelven el problema de reconocimiento de las instancias CoE en forma normal conjuntiva.

En 2004, Benhamou [ben04] establece y desarrolla una relación entre el problema de SATISFACIBILIDAD PROPOSICIONAL (SAT) y los grafos bipartitos. A partir de esta relación, este autor presenta una clase de satisfacibilidad resoluble en tiempo polinomial, que incluye la clase r-r-SAT. Las instancias de r-s-SAT [tov84] son CNF-fórmulas cuyas instancias están formadas por cláusulas con r literales de modo que cada variable aparece a lo máximo en s cláusulas.

Queda como problema abierto establecer si existe alguna relación entre estas clases polinomiales.

1.8. Algoritmos polinomiales para problemas SAT

Dado que SAT es un problema NP-completo, sucede que no es conocido ningún algoritmo determinista polinomial que lo resuelva (y que no existe a menos que P=NP). En este contexto, la obtención de algoritmos

eficientes ha sido objetivo de muchos investigadores en este dominio.

Una de las primeras máquinas para la resolución de un problema lógico fue desarrollada por el lógico y economista William Stanley Jevons (1835-1882) que, en 1869, construyó el primer modelo de la *Jevon's Logic Machine* [jev90]. Este notable aparato fue el primero que permitía resolver un problema lógico más rápidamente de lo que se podría resolver en la época sin el uso de cualquier máquina. Jevons era un aficionado a la lógica booleana y su máquina era una mezcla de un ábaco lógico con un piano (siendo incluso conocida como *piano lógico*). Este artefacto, con cerca de 1 metro de altura, consistía en un conjunto de teclas, palancas y ruedas, unidas a letras que podían quedar visibles u ocultas. Cuando el operador presionaba teclas representando operaciones lógicas, la letra apropiada aparecía revelando el resultado correcto[1].

Sin embargo, es con el advenimiento del computador cuando el tema adquiere amplio interés, especialmente en la década de los 60 del siglo pasado. Algoritmos e implementaciones como Davis—Putnam [DP60], DPLL [DLL62], diagramas binarios de decisión [bry86], GSAT y WSAT [SLM92], Stalmarck [SS00], GRASP [SS96, SS99], Chaff [MMZ01], SATO [ZS96, zha97] y Berkmin [GN02] constituyen algunas de las metodologías de abordaje para la resolución del problema de satisfacibilidad.

A pesar de alguna desactualización (no en vano el trabajo tiene una década en un campo de investigación activo), Gu, Purdom, Franco y Wah (1997) [GPF97] presenta un abordaje sistemático de los distintos tipos de estrategias y algoritmos para el problema SAT. Este trabajo incluye una sección dedicada a las clases de satisfacibilidad polinomiales con referencia a algunos algoritmos deterministas polinomiales para su resolución.

Dedicamos la presente sección a la presentación de algoritmos deterministas polinomiales que permiten la resolución de tres de las bien conocidas clases polinomiales, 2SAT, SLUR y LinAut. La primera por el interés que presenta en sí misma y las otras dos por contener a todas las demás clases polinomiales bien conocidas (v. Figura 1.17).

Directamente aplicados o bien desarrollados para la resolución de instancias de clases de satisfacibilidad polinomiales se pueden encontrar varios algoritmos y estrategias.

Para la clase 2SAT destacamos las aproximaciones al problema de de Even, Itai y Shamir (1976) [EIS76], que involucra computación paralela, las de Tarjan (1979) [APT79], Chandru *et al.* (1990) [CCH90] y Pretolani [pre93], basados en distintas formas de grafos o hipergrafos, y las de Dalal y Etherington (1992) [DE92] y Del Val (2000) [val00], que exploran propiedades de cláusulas y literales.

[1] http://www.rutherfordjournal.org/article010103.html

El algoritmo SLUR *(Single Lookahead Unit Resolution)* fue desarrollado por Schlipf, Annexstein, Franco y Swaminathan (1995) con el objetivo de resolver las instancias de la clase *extended* Horn sin necesidad de su reconocimiento previo, para lo cual no se conoce ningún algoritmo polinomial [SAF95]. Este algoritmo vino a dar su nombre a la clase de satisfacibilidad polinomial SLUR que contiene propiamente a las clases Horn, *renamable* Horn, *extended* Horn y CC-Balanced [FG03]. Basada en los conceptos de *asignación de autarquía* y de *autarquía lineal*, la clase LinAut asume especial interés pues no es comparable con la clase SLUR [maa00] y entre ambas clases contienen todas a las demás clases de satisfacibilidad polinomiales bien conocidas.

§ 1.8.1. Algoritmos para 2SAT.— La mayoría de los algoritmos utilizados en la resolución del problema 2SAT se basan en el concepto de *resolución unitaria*. Si existe una cláusula unitaria, a su literal l se le atribuye el valor 1. Todas las cláusulas que contengan este literal son verdaderas y son eliminadas de la fórmula proposicional. El literal complementario $\bar{l}$ se elimina de cada una de las cláusulas que lo contenga. El procedimiento se repite mientras exista alguna cláusula unitaria. Si durante el proceso el único literal de una cláusula unitaria fuese eliminado, se obtiene una cláusula nula que no es satisfacible por ninguna atribución de verdad y la fórmula proposicional no es satisfacible.

El algoritmo de propagación unitaria, PROPUNIT que se presenta a continuación es una versión adaptada de la propuesta por Dalal y Etherington que admite una implementación en tiempo lineal respecto al tamaño de la entrada, $O(|\mathcal{F}|)$ [DE92] (v. Algoritmo 1).

Algoritmo 1 PROPUNIT($\mathcal{F}$) (Propagación Unitaria [DE92]).

Entrada: Fórmula proposicional $\mathcal{F}$
Salida: Fórmula proposicional $\mathcal{F}$ y una atribución de verdad τ

```
    τ = ∅
    while existan cláusulas unitarias en F do
      selecciona una cláusula unitaria C = [l]
      τ = τ ∪ {l}
 5:   F = {C ∈ F : l ∉ C}
      F = {C − [l̄] : C ∈ F}
      if □ ∈ F then
        F = {□}
      end if
10: end while
    return (F,τ)
```

Una implementación más eficiente suspende la operación de sim-

plificación de las cláusulas centrándose en la resolución, operación que consiste en eliminar el complementario de un literal verdadero. En el momento de eliminar un literal de una cláusula, si esta contiene un literal atribuido con el valor verdadero, entonces la cláusula es eliminada de la fórmula; en caso contrario se efectúa la resolución. Una descripción detallada de esta implementación puede consultarse en el trabajo de Zhang y Stickel [ZS96].

Consideremos una fórmula proposicional 2SAT, $\mathcal{F}$, sin cláusulas unitarias. Para un literal en una de sus cláusulas, $l \in C$, consideremos $(\mathcal{F}^1, \tau) = \text{PROPUNIT}(\mathcal{F} \cup \{[l]\})$. Se verifica que $\mathcal{F}^1 \subsetneqq \mathcal{F}$, pues por lo menos la cláusula C ha sido eliminada de la fórmula inicial y $\mathcal{F}^1$ no contiene cláusulas unitarias. Si $\mathcal{F}^1$ no contiene la cláusula nula, $\mathcal{F}$ es satisfacible si y sólo si $\mathcal{F}^1$ es satisfacible, por lo que podemos repetir el procedimiento. En el caso de que $\mathcal{F}^1$ contenga la cláusula nula, se aplica nuevamente el algoritmo de propagación unitaria pero asumiendo que el literal complementario $\bar{l}$ es verdadero, $(\mathcal{F}^2, \tau) = \text{PROPUNIT}(\mathcal{F} \cup \{[\bar{l}]\})$. La fórmula proposicional $\mathcal{F}$ no es satisfacible si y sólo si, en algún paso, tanto $\mathcal{F}^1$ como $\mathcal{F}^2$ contienen la cláusula nula.

Para uno de los algoritmos más eficientes para la resolución de las instancias de la clase 2SAT, recurriendo al algoritmo PROPUNIT, véase el trabajo de Del Val (2000) [val00].

§ 1.8.2. Algoritmo para SLUR.— El algoritmo SLUR (v. Algoritmo 2), que da el nombre a la clase homónima, fue desarrollado para la resolución de las entradas de una clase de satisfacibilidad que contenía (propiamente) a las clases polinomiales *extended* Horn y *CC-balanced*. Uno de los principales objetivos fue diseñar un algoritmo completo para las fórmulas de las clases referidas sin necesidad de identificar previamente a que clase pertenecen cada una de las entradas. Esto es debido a que no se conoce ningún algoritmo polinomial que efectúe el reconocimiento de las cláusulas de la clase *extended* Horn.

El algoritmo se inicia ejecutando el algoritmo PROPUNIT y obteniendo así una fórmula sin cláusulas unitarias. En el caso de que ésta contenga la cláusula nula, el algoritmo termina con el mensaje *no satisfacible*. Caso contrario, si la fórmula está en SLUR, es satisfacible. En cada iteración, escogiendo arbitrariamente una variable v, la fórmula es resuelta con el literal afirmado, v, y con el literal negado, $\overline{v}$, utilizando el algoritmo de propagación unitaria. Si en ambas ramas se obtiene una fórmula con la cláusula nula, el algoritmo termina con el mensaje *desisto*. En el caso de algún ramo *falsifique* la fórmula, el algoritmo continúa con la fórmula obtenida en la otra rama. Si en ninguna de las ramas la fórmula fuese falsificada, el algoritmo continua escogiendo arbitrariamente una de las ramas hasta que que no existan más variables. De

Algoritmo 2 SLUR($\mathcal{F}$) (Single Lookahead Unit Resolution [SAF95, FG03]).

Entrada: Fórmula proposicional $\mathcal{F}$
Salida: Asignación de verdad τ o una de las expresiones *no satisfacible* o *desisto*

```
(F, τ)=PROPUNIT(F)
if □ ∈ F then
  return no satisfacible
else
  while F ≠ ∅ do
    selecciona una variable v ∈var(F)
    (F^1, τ^1)=PROPUNIT(F, {[v̄]})
    (F^2, τ^2)=PROPUNIT(F, {[v]})
    if □ ∈ F^1 y □ ∈ F^2 then
      return desisto
    else
      if □ ∈ F^1 then
        (F, τ)=(F^2, τ ∪ τ^2)
      else if □ ∈ F^2 then
        (F, τ)=(F^1, τ ∪ τ^1)
      else
        escoge arbitrariamente una de las dos opciones:
        1. (F, τ)=(F^1, τ ∪ τ^1)
        2. (F, τ)=(F^2, τ ∪ τ^2)
      end if
    end if
  end while
end if
return τ
```

este modo, en cada paso, se amplía la atribución de verdad, obteniéndose un modelo para la fórmula proposicional.

El algoritmo SLUR se ejecuta en tiempo lineal si se sigue la modificación propuesta por Truemper (1998) [tru98], que consiste en efectuar la propagación unitaria para las dos ramas en simultáneo, abandonando una de las ramas si la otra termina sin producir la cláusula nula.

El problema del reconocimiento de las instancias de la clase SLUR está por resolver. Recuérdese que una instancia pertenece a la clase SLUR si, para toda la secuencia de variables el algoritmo SLUR nunca termina con el mensaje *desisto*.

Sin embargo, este algoritmo puede ser aplicado a cualquier fórmula proposicional, independientemente de si pertenece o no a la clase. Así, si la respuesta devuelta es el mensaje *no satisfacible* la fórmula pertenece

a SLUR y, si la respuesta es el mensaje *desisto*, la fórmula no pertenece a SLUR. En caso contrario, la fórmula es satisfacible, pero no se sabe si es o no una instancia de esta clase.

§ 1.8.3. Algoritmo para LinAut.— Dada una fórmula proposicional $\mathcal{F}$ y su matriz cláusula-variable, M, se dice que $\mathcal{F}$ admite una *autarquía lineal* $\mathbf{x} \in \mathbb{Q}^n$ si

$$\begin{cases} M\mathbf{x} \geq 0 \\ \mathbf{x} \neq 0 \end{cases} \tag{1.3}$$

Si existe una *autarquía lineal*, $\mathbf{x} \in \mathbb{Q}^n$, existe también una *asignación de autarquía* τ de modo que $v_j \in \tau$ si $sgn(x_j) = 1$ y $\overline{v_j} \in \tau$ si $sgn(x_j) = -1$.

Para una cláusula afectada por la atribución τ existe una variable $x_j \neq 0$ tal que que $M_{i,j} \neq 0$. Como

$$\sum_j M_{i,j}x_j = \sum_{j,x_j \neq 0} M_{i,j}x_j \geq 0$$

no todos los sumandos pueden ser negativos. Por lo que, para algún j, $sgn(x_j) = M_{i,j}$, y por lo tanto, la cláusula C_i es verdadera para la atribución (parcial) τ.

Dado que todas las cláusulas de $\mathcal{F}$ afectadas por la atribución τ son verdaderas (para esta atribución parcial), eliminándolas, se obtiene una subfórmula $\mathcal{F}^1 \subset \mathcal{F}$, definida sobre las variables para las cuales $x_j = 0$. En estas condiciones, una *autarquía lineal*, induce una descomposición no trivial de la fórmula proposicional involucrada. Por otro lado, $\mathcal{F}$ es satisfacible si y sólo si $\mathcal{F}^1$ es satisfacible.

Utilizando algoritmos de programación lineal es posible determinar si existen o no *autarquías lineales* y calcularlas, en su caso.

Una fórmula proposicional sin cláusulas unitarias, $\mathcal{F}$, se denomina que pertenece a la clase de satisfacibilidad polinomial LinAut si aplicando sucesivamente la descomposición inducida por las *autarquías lineales*, se obtiene o una subfórmula vacía o una subfórmula 2SAT. En el primer caso $\mathcal{F}$ es satisfacible y en el segundo no satisfacible [kul00, maa00].

CAPÍTULO 2

PURL

En el capítulo anterior citamos algunas de las clases polinomiales de satisfacibilidad conocidas. De hecho, se citaron las que Franco y Gelder designaron como *bien conocidas (well-known)*. Entre éstas, debemos destacar 2SAT, la paradigmática clase polinomial de satisfacibilidad, SLUR y LinAut. Estas dos últimas, no comparables entre si, en conjunto contienen todas las demás clases polinomiales de satisfacibilidad bien conocidas (v. Figura 1.17). Las instancias 2SAT son fácilmente reconocibles, pues cada una de sus cláusulas contiene como mucho dos literales. El reconocimiento de las instancias LinAut se hace mediante un algoritmo polinomial, a partir de las propiedades del espacio de soluciones. Las instancias SLUR se caracterizan a partir del comportamiento de un algoritmo con una componente no determinista y no se conoce ningún algoritmo que en tiempo polinomial reconozca su pertenencia a SLUR.

Explorando la existencia de bloqueos locales en la satisfacibilidad de algunas fórmulas proposicionales y basándonos en el (nuevo) concepto de *literal eliminable*, desarrollaremos una nueva clase de satisfacibilidad polinomial que designaremos por PURL (*PropUnit Removable Literal*) y que presentaremos en este capítulo. Para la resolución de sus instancias se desarrolla el algoritmo ELIMINALITERALES, el cual permite identificar y excluir cierto tipo de literales eliminables de cada una de las cláusulas en tiempo polinomial.

La identificación de la clase PURL fue consecuencia de los trabajos desarrollados para la resolución del problema de emparejamiento ortogonal simple en el cilindro (EOSC) y en el toro (EOST). Reduciendo el problema geométrico EOSC al problema de satisfacibilidad, se obtiene una clase de fórmulas proposicionales, EOSC-SAT, que constituye una de las subclases de 2PURL.

Intentando relacionar la nueva clase de satisfacibilidad que se propone con las clases conocidas anteriormente, se constata que cada una de las clases polinomiales de satisfacibilidad bien conocidas son subclases de PURL. Queda abierto el problema de determinar si PURL podría ser la superclase de satisfacibilidad polinomial conjeturada por Gu, Purdom, Franco y Wah [GPF97] y por Franco y Gelder [FG03].

2.1. Definiciones

§ 2.1.1. Literal eliminable.— Un literal, en una cláusula de una fórmula proposicional dada, es *eliminable* si puede ser excluido sin alterar el espacio de soluciones de satisfacibilidad (modelos) de la fórmula. Pero antes de precisar el concepto, comencemos por establecer algunas definiciones previas necesarias.

Dada una fórmula proposicional $\mathcal{F}$, para cada cláusula $C \in \mathcal{F}$ denotamos $T_1(C)$ al conjunto de las atribuciones de verdad para las cuales un literal de C tiene el valor verdadero y los restantes literales de C tienen valor falso. Siendo l un literal de la cláusula C, $T_1(l, C)$ designa el subconjunto de las atribuciones de verdad de $T_1(C)$ para las cuales el literal l es verdadero.

El conjunto $T_1(l, C)$ constituye una variante de la definición de 1-vecindad de una atribución de verdad propuesta por Goldberg [gol02, gol05]. En estos trabajos, el autor desarrolla el concepto de *subconjunto de atribuciones estable* (SAE) y un algoritmo no determinista polinomial para determinar la existencia de un SAE. Para una fórmula proposicional, la existencia de un SAE equivale a la no satisfacibilidad de esa fórmula.

Una fórmula proposicional satisfacible, que no sea una tautología, admite como solución una atribución de verdad perteneciente a uno de los conjuntos $T_1(l, C)$, para una de sus cláusulas, conforme se sigue del siguiente resultado debido a Goldberg.

Teorema 2.1 ([gol02])**.** *Sea $\mathcal{F}$ una fórmula proposicional satisfacible. Si $\mathcal{F}$ no es una tautología, entonces existe una cláusula $C \in \mathcal{F}$ y una atribución de verdad $\tau \in T_1(l, C)$, para algún literal $l \in C$, tal que $\mathcal{F}(\tau) = 1$.*

De ahora en adelante, y siempre que nada se refiera en contrario, consideraremos que cada una de las fórmulas proposicionales en el texto no constituye una tautología.

Dada una fórmula proposicional $\mathcal{F}$, si para un literal $l \in C$ ninguna de las atribuciones del conjunto $T(l, C)$ satisface $\mathcal{F}$, entonces cualquier modelo de $\mathcal{F}$ contiene algún literal de C distinto de l. Por tanto, para este modelo la cláusula $C - [l]$ es verdadera. Lo que motiva la siguiente definición de *literal eliminable*.

Definición 2.1. Dada una fórmula proposicional $\mathcal{F}$ y una de sus cláusulas C, diremos que el literal $l \in C$ es eliminable si, para cualquiera de las atribuciones de verdad $\tau \in T_1(l, C)$, $\mathcal{F}(\tau) = 0$.

Excluyendo este literal de la correspondiente cláusula, C, se obtiene una nueva fórmula proposicional, $\mathcal{F}^{'} = (\mathcal{F} - \{C\}) \cup \{C^{'}\}$, dónde $C^{'} = C - [l]$. De acuerdo al siguiente resultado, las fórmulas proposicionales, $\mathcal{F}$ y $\mathcal{F}^{'}$ son lógicamente equivalentes.

Teorema 2.2. *Sean $\mathcal{F}$ una fórmula proposicional, $C \in \mathcal{F}$ una cláusula y $l \in C$ un literal eliminable. Entonces $\mathcal{F}^{'} = (\mathcal{F} - C) \cup (\{C - [l]\})$ es lógicamente equivalente a $\mathcal{F}$.*

Demostración. Dado que dos fórmulas son lógicamente equivalentes si tienen los mismos modelos, comencemos por suponer que τ es un modelo para la fórmula simplificada, $\mathcal{F}^{'}(\tau) = 1$. Como $\mathcal{F}$ y $\mathcal{F}^{'}$ coinciden en todas las cláusulas excepto en la cláusula C, $\mathcal{F}^{'} - C^{'} = \mathcal{F} - C$ y $C^{'} \subset C$, todas las cláusulas de $\mathcal{F}$ son satisfechas por la atribución τ, $\mathcal{F}(\tau) = 1$, por lo que τ es también un modelo de $\mathcal{F}$.

Recíprocamente, supongamos que $\mathcal{F}(\tau) = 1$ y, por reducción al absurdo, que $\mathcal{F}^{'}(\tau) = 0$. Entonces, todos los literales de la cláusula $C^{'}$ son necesariamente falsos y $l \in C$ es un literal verdadero para la atribución τ. Por lo que $\tau \in T_1(l, C)$ y $\mathcal{F}(\tau) = 1$, lo que está en contradicción con la hipótesis de que $l \in C$ sea un literal eliminable ($l \in C$ es eliminable si para cualquiera asignación $\tau \in T(l, C)$, $\mathcal{F}(\tau) = 0$). □

La relación lógicamente equivalente entre fórmulas proposicionales constituye una relación de equivalencia pues es reflexiva, simétrica y transitiva. Por lo que excluyendo cada uno de los literales eliminables se obtiene una fórmula proposicional lógicamente equivalente a la primera. En el caso de que $\mathcal{F}$ no sea satisfacible, todos los literales son eliminables por lo que esta fórmula proposicional es lógicamente equivalente a la fórmula con la cláusula nula, $\mathcal{F} = \{\square\}$.

En el caso particular de la clase 2SAT, a partir de cualquiera de los algoritmos deterministas polinomiales utilizados en su resolución [DE92, val00], para cada instancia es posible encontrar una fórmula proposicional lógicamente equivalente en tiempo polinomial, detectando y eliminando cada uno de los literales eliminables. Por lo tanto, para una cláusula $[u, v] \in \mathcal{F}$, el literal u es eliminable si y sólo si la fórmula proposicional $\mathcal{F} \cup \{[u], [\overline{v}]\}$ no es satisfacible.

En cambio, para una fórmula proposicional 3SAT, determinar si cada uno de los literales de sus cláusulas es eliminable equivale a determinar su no satisfacibilidad. Por lo que, como el problema de satisfacibilidad de la clase 3SAT es intratable (i.e. está en NP) entonces saber si una cláusula contiene literales eliminables es igualmente un problema intratable.

Literal p-eliminable

Es un hecho conocido que la no satisfacibilidad de las fórmulas proposicionales puede ser obtenida, en algunos casos, a partir de procedimientos de búsqueda local. El concepto de literal *p-eliminable*, que se define a continuación, permite identificar algunas formas de literales eliminables locales, en tiempo polinomial.

Considerando una cláusula proposicional no nula $C \in \mathcal{F}$ y uno de sus literales $l \in C$, denotamos $C_1 = Flip(C, l)$ a la cláusula que contiene los mismos literales que C sustituyendo el literal l por el negado $\bar{l}$. Por ejemplo, siendo $C = [l, x, y]$, $Flip(C, l) = [\bar{l}, x, y]$.

Teniendo presente que una cláusula C es una *consecuencia lógica* de una fórmula proposicional $\mathcal{F}$ ($\mathcal{F} \models C$) si todo modelo de $\mathcal{F}$ es también un modelo de C, podemos enunciar el siguiente resultado:

Lema 2.3. *Dada una fórmula proposicional $\mathcal{F}$ y un literal de una de sus cláusulas, $l \in C$, este literal es eliminable si y sólo si $Flip(C, l)$ es una consecuencia lógica de $\mathcal{F}$.*

Demostración. Si suponemos que $l \in C$ es un literal eliminable entonces, por definición, $\mathcal{F}(\tau) = 0$, para cualquier atribución $\tau \in T_1(l, C)$. Si τ^1 es un modelo para $\mathcal{F}$, entonces $\mathcal{F}(\tau^1) = 1$ y, en particular, $C(\tau^1) = 1$. Por tanto, existe un literal $u \in C - [l]$ verdadero para la atribución τ^1 y, por lo tanto, como $u \in Flip(C, l)$, $Flip(C, l)(\tau^1) = 1$, por lo que la cláusula $C_1 = Flip(C, l)$ es una consecuencia lógica de $\mathcal{F}$.

A la inversa, si admitimos que $C_1 = Flip(C, l)$ es una consecuencia lógica de $\mathcal{F}$. Sea $\tau \in T_1(l, C)$ una atribución de verdad. Por lo tanto, $l \in \tau$ y por cada literal $u \in C - [l]$, su complementario pertenece a τ ($\overline{C - [l]} \subset \tau$). Pero en estas condiciones, la cláusula $Flip(C, l)$ es falsa para la atribución τ y, como $Flip(C, l)$ es una consecuencia lógica de $\mathcal{F}$, $\mathcal{F}(\tau) = 0$. Lo que permite concluir que el literal $l \in C$ es un literal eliminable. □

Para facilitar la notación, denotaremos $\overline{C}$ al conjunto formado por los complementarios de los literales de C. Por ejemplo, para la cláusula $C = [x, y, w]$ se tiene $\overline{C} = \{\overline{x}, \overline{y}, \overline{w}\}$.

Para una cláusula $C \in \mathcal{F}$, si la cláusula $Flip(C, l)$ es una consecuencia lógica de $\mathcal{F}$, entonces cualquier atribución de verdad τ que

contenga el conjunto de literales $\overline{Flip(C,l)}$ falsifica la fórmula $\mathcal{F}$, i.e. $\mathcal{F}(\tau) = 0$. Por lo que, si la propagación unitaria de la atribución de verdad $\tau = \overline{Flip(C,l)}$ falsifica $\mathcal{F}$, la cláusula $Flip(C,l)$ es una consecuencia lógica de $\mathcal{F}$ y, usando el lema anterior, $l \in C$ es un literal eliminable.

A modo de ejemplo consideremos la fórmula proposicional, en la cual las variables (y sus correspondientes literales) son representadas por los respectivos índices, como es tradicional en la literatura. La negación del literal es representada por el signo menos ($-$).

$$\begin{aligned}\mathcal{F} = \quad & \{[1,2,-4],[-1,2,4],[2,-3,-4],[-2,3,-4],\\ & [1,2,6],[-1,2,-6],[2,-5,-6],[-2,5,-6],\\ & [3,4,6],[-3,4,-6],[4,5,-6],[-4,-5,-6]\}\end{aligned} \tag{2.1}$$

Considerando la cláusula $[1,2,-4] \in \mathcal{F}$, se comprueba fácilmente que la atribución de verdad obtenida por propagación unitaria de la atribución (parcial) $\overline{Flip([1,2,-4],-4)} = \{-4,-1,-2\}$ falsifica la fórmula $\mathcal{F}$. Por lo tanto, el literal $-4 \in [1,2,-4]$ es eliminable. Lo que implica que la cláusula $[1,2,-4]$ puede ser substituida por la cláusula $[1,2]$, obteniéndose así una fórmula proposicional lógicamente equivalente.

Repitiendo el procedimiento para cada una de las cláusulas y literales sucesivos de $\mathcal{F}$ se obtiene la fórmula $\mathcal{F}^E$, lógicamente equivalente a $\mathcal{F}$.

$$\begin{aligned}\mathcal{F}^E = \quad & \{[1,2],[2,4],[2,-3],[-2,3,-4],\\ & [1,2],[2,-6],[2,-6],[5,-6],\\ & [3,4,6],[-3,-6],[5,-6],[-4,-6]\}\end{aligned} \tag{2.2}$$

Es evidente que no todos los literales eliminables pueden ser identificados según el proceso ilustrado por el ejemplo anterior, el cual presenta como una de sus ventajas principales la simplicidad. I.e., la sistemática aplicación de este proceso constituye un procedimiento incompleto para la determinación de los literales eliminables. Los literales eliminables que puedan ser determinados por este proceso son denominados *literales p-eliminables* y su identificación puede ser realizada en tiempo polinomial utilizando propagación unitaria.

Por compatibilidad con las entradas del algoritmo de propagación unitaria, definimos el operador $\mathcal{U}$ aplicado a un conjunto de literales o a una atribución de verdad de forma que devuelve una fórmula proposicional con todas las cláusulas unitarias conteniendo cada una de ellas uno de los literales de la referida atribución. Por ejemplo $U(\{l_1,\overline{l_2}\}) = \{[l_1],[\overline{l_2}]\}$. Usando este operador podemos definir formalmente el concepto de *literal p-eliminable*:

Definición 2.2. Sea $\mathcal{F}$ una fórmula proposicional y C una de sus cláusulas. El literal $l \in C$ se llama p-eliminable si el algoritmo de propagación

unitaria aplicado a $\mathcal{F} \cup \mathcal{U}(\overline{Flip(C,l)})$ devuelve una fórmula proposicional que contiene la cláusula nula.

Es una afirmación obvia que un literal p-eliminable es eliminable, por lo que que excluyendo cada uno de los literales p-eliminables de las respectivas cláusulas se obtiene una fórmula proposicional lógicamente equivalente. Dado que, con la implementación de Dalal y Etherington [DE92], el algoritmo PROPUNIT corre en tiempo lineal respecto al tamaño de la fórmula $|\mathcal{F}|$, para una cláusula dada es posible determinar si un de sus literales es p-eliminable en tiempo lineal $O(|\mathcal{F}|)$ y eliminarlo en tiempo constante.

Consideremos otro ejemplo, la siguiente fórmula proposicional:

$$\mathcal{F} = \quad \{[l_1, l_2], [l_2, l_3, l_4], [l_2, \overline{l_3}, l_4], [l_2, l_3, \overline{l_4}], [l_2, \overline{l_3}, \overline{l_4}]\}$$

El literal $l_3 \in [l_2, l_3, l_4]$ es p-eliminable (en $\mathcal{F}$) y excluyéndolo se obtiene la fórmula proposicional lógicamente equivalente

$$\mathcal{F}' = \{[l_1, l_2], [l_2, l_4], [l_2, \overline{l_3}, l_4], [l_2, l_3, \overline{l_4}], [l_2, \overline{l_3}, \overline{l_4}]\}$$

Fijémonos que el literal $l_1 \in [l_1, l_2]$ que no era p-eliminable en $\mathcal{F}$ es ahora p-eliminable en $\mathcal{F}'$. Literales en estas condiciones son denominados literales p-eliminables ocultos. Para detectar y excluir literales p-eliminables (incluyendo los que llamamos p-eliminables ocultos) se ha elaborado el algoritmo ELIMINALITERALES($\mathcal{F}$) (v. Algoritmo 3).

Algoritmo 3 ELIMINALITERALES($\mathcal{F}$)

Entrada: Fórmula proposicional $\mathcal{F}$ (sin cláusula nula)
Salida: Fórmula sin literales p-eliminables, $\mathcal{F}$

```
    repeat
      remlitfree = true
      for all C ∈ F do
        for all l ∈ C do
5:        (F', τ') = PROPUNIT(F ∪ U(¬Flip(C, l)))
          if □ ∈ F' then
            F = (F − {C}) ∪ ({C − [l]}), remlitfree = false
          end if
          if □ ∈ F then
10:         F = {□}
          end if
        end for
      end for
    until F = {□} OR remilitfree = true
15: return (F)
```

La búsqueda de literales eliminables se hace de una forma secuencial para cada una de las cláusulas $C_1, C_2, \cdots, C_m$ de $\mathcal{F}$. En cada cláusula, se determina la propagación unitaria de cada atribución $\overline{Flip(C_i, l_{i_j})}$ en $\mathcal{F}$ para cada literal $l_{i_j} \in C_i$. Si l_{i_j} es un literal p-eliminable, se actualiza la cláusula eliminándolo ($C_i = C_i - [l_{i_j}]$). Naturalmente, esto implica una actualización de la fórmula proposicional, i.e. $\mathcal{F} = (\mathcal{F} - \{C_i\}) \cup (\{C_i \setminus [l_{i_j}]\})$.

El algoritmo finaliza cuando se obtiene una cláusula nula o una fórmula proposicional lógicamente equivalente a $\mathcal{F}$ sin literales p-eliminables.

Si consideramos que el algoritmo de propagación unitaria PROPUNIT se ejecuta en tiempo lineal respecto al tamaño de la fórmula proposicional, $O(|\mathcal{F}|)$, se tiene que en el caso de que no exista ningún literal p-eliminable el algoritmo corre en tiempo cuadrático $O(|\mathcal{F}|^2)$. El peor caso es que se elimine un único literal en cada iteración sobre las cláusulas y literales de $\mathcal{F}$ y entonces, el algoritmo tiene complejidad $O(|\mathcal{F}|^3)$.

Lema 2.4. *Para una* CNF*-fórmula proposicional $\mathcal{F}$, el algoritmo* ELIMINALITERALES$(\mathcal{F})$ *corre en tiempo cúbico relativamente al tamaño de la fórmula, $O(|\mathcal{F}|^3)$, en peor caso.*

El algoritmo ELIMINALITERALES$(\mathcal{F})$ es sensible a las entradas pues la ordenación de los literales y cláusulas de una fórmula proposicional puede influenciar el resultado. Para ilustrar esta afirmación consideremos la fórmula proposicional:

$$\mathcal{F} = \{[1, 2, 3], [2, 4], [3, \overline{4}], [1, \overline{5}], [\overline{2}, 5]\} \tag{2.3}$$

Como se comprueba fácilmente, los literales $1 \in [1, 2, 3]$ y $2 \in [1, 2, 3]$ son ambos p-eliminables en $\mathcal{F}$. Sin embargo, dependiendo del orden de eliminación, podemos obtener dos fórmulas proposicionales distintas $\mathcal{F}^1$ o $\mathcal{F}^2$:

$$\mathcal{F}^1 = \{[2, 3], [2, 4], [3, \overline{4}], [1, \overline{5}], [\overline{2}, 5]\} \tag{2.4a}$$

$$\mathcal{F}^2 = \{[1, 3], [2, 4], [3, \overline{4}], [1, \overline{5}], [\overline{2}, 5]\}. \tag{2.4b}$$

Cada una de las fórmulas, $\mathcal{F}^1$ y $\mathcal{F}^2$, es lógicamente equivalente a $\mathcal{F}$ y, por lo tanto, lógicamente equivalente a la otra. Esta situación solo puede ocurrir se existe una cláusula con dos literales p-eliminables. El siguiente resultado muestra que un literal $l \in C$ p-eliminable en $\mathcal{F}$ es también p-eliminable en las fórmulas $\mathcal{F}^1$ obtenidas de $\mathcal{F}$ excluyendo sucesivamente literales p-eliminables de otras cláusulas, siempre que la cláusula C no sea modificada.

Lema 2.5. *Consideremos una cláusula $C \in \mathcal{F}$ y sea $\mathcal{F}^E$ una fórmula lógicamente equivalente a $\mathcal{F}$ obtenida por exclusión de literales p-eliminables. Si la misma cláusula pertenece también a $\mathcal{F}^E$, $C \in \mathcal{F}^E$, y*

$l \in C$ es un literal p-eliminable en $\mathcal{F}$, entonces l es también p-eliminable en $\mathcal{F}^E$.

Demostración. Sea $\tau^0 = \overline{Flip(C, l)}$ y consideremos la propagación unitaria de esta atribución de verdad en $\mathcal{F}$ y en $\mathcal{F}^E$, $(\mathcal{F}^1, \tau^1) = \text{PROPUNIT}(\mathcal{F} \cup \mathcal{U}(\tau^0))$ y $(\mathcal{F}^2, \tau^2) = \text{PROPUNIT}(\mathcal{F}^E \cup \mathcal{U}(\tau^0))$, respectivamente.

Para que algún literal sea añadido a la asignación τ^0, por propagación unitaria de $\mathcal{F} \cup \mathcal{U}(\tau^0)$, es condición necesaria y suficiente que exista una cláusula $D \in \mathcal{F}$ y un literal $u \in D$ tal que el complementario de cada uno de los literales de $D - [u]$ pertenezca a τ^0, i.e., $\overline{D - [u]} \subset \tau^0$. Sea $\tau^{0,1} = \tau^0 \cup \{u\}$. Entonces, designando por $D^{'}$ a la cláusula de $\mathcal{F}^E$ correspondiente a D, $D^{'} \subset D$. Por lo tanto, como $\overline{D^{'} - [u]} \subset \tau^0$, o bien sucede que $u \in D^{'}$ y en la propagación de $\mathcal{F}^E \cup \mathcal{U}(\tau^0)$ y $\tau^{0,2} = \tau^0 \cup \{u\}$ o bien que $u \notin D^{'}$ y, en este caso, el literal $l \in C$ es p-eliminable en $\mathcal{F}^E$. Supongamos que este ultimo caso no ocurre.

Repitiendo el argumento anterior, siempre que se añade un literal a la asignación $\tau^{0,1}$, lo mismo literal de añade a $\tau^{0,2}$, por tanto $\tau^{0,1} \subset \tau^{0,2}$.

Como, en algún paso, existe una cláusula $E \in \mathcal{F}$ con todos sus literales falsos para la asignación $\tau^{0,1}$, para la cláusula $E^{'} \in \mathcal{F}^E$, correspondiente a E se verifica la inclusión $E^{'} \subset E$ y, por lo tanto, $\overline{E^{'}} \subset \overline{E} \subset \tau^{0,1} \subset \tau^{0,2} \subset \tau^2$. Se concluye, por tanto, que el literal $l \in C$ es p-eliminable en $\mathcal{F}^E$. □

El algoritmo ELIMINALITERALES($\mathcal{F}$) presentado es susceptible de incorporar algunas mejoras de eficiencia. En particular, antes de iniciar el procedimiento para detectar literales p-eliminables y siempre que por eliminación de literales se obtenga una cláusula unitaria, el algoritmo de propagación unitaria puede ser invocado para simplificación de la fórmula. Cada vez que el algoritmo de propagación unitaria es ejecutado (en la línea 5), si la fórmula retornada $\mathcal{F}^{'}$ no contiene la cláusula nula y $\mathcal{F}^{'} \subset \mathcal{F}$ las fórmulas $\mathcal{F}^{'}$ y $\mathcal{F}$ son equisatisfacibles, por lo que el procesamiento puede proseguir con la fórmula proposicional $\mathcal{F}^{'}$. Naturalmente que el resultado obtenido, $(\mathcal{F}^{'})^E$ es una fórmula equisatisfacible con la instancia de entrada.

Sin embargo, la ganancia principal consiste en que, para una misma cláusula C y un literal $l \in C$, se evita ejecutar repetidas veces el algoritmo de propagación unitaria. Estudiaremos más detenidamente esta mejora en la Sección 2.7, al final del presente capítulo, dónde presentaremos una algoritmo que permite, en muchos casos, detectar y eliminar literales p-eliminables en tiempo cuadrático.

§ 2.1.2. PURL, una nueva clase de satisfacibilidad polinomial .— A partir del concepto de literal p-eliminable definimos a continuación una nueva clase de satisfacibilidad polinomial que designamos por PURL

(PropUnit Removable Literal). Su definición se hace en términos de un algoritmo polinomial incompleto que permite determinar la no satisfacibilidad. En general, determinar si una instancia pertenece a PURL o no es un problema abierto.

Como se ha referido anteriormente, se conocen varias clases de satisfacibilidad resolubles por algoritmos deterministas polinomiales. Algunas de estas clases se caracterizan a partir de ciertas propiedades o características sintácticas de sus cláusulas, e.g. 2SAT y Horn; otras a partir de propiedades del correspondiente espacio de soluciones, LinAut y otras mediante el comportamiento de un algoritmo, SLUR.

En este sentido, la definición que se propone para la clase de satisfacibilidad PURL es diferente, pues su caracterización se hace a partir a partir de sus subclases (propias) y del algoritmo polinomial ELIMINALITERALES($\mathcal{F}$) que permite la identificación y exclusión de literales p-eliminables de la fórmula proposicional.

Definición 2.3. Sea Ω una clase de fórmulas proposicionales. Ω es una subclase de PURL si y sólo si para cada fórmula proposicional $\mathcal{F} \in \Omega$, no satisfacible, $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$ contiene la cláusula nula, para cualquiera ordenación de sus cláusulas y literales.

Como primera consecuencia de la definición tenemos que si Ω es una subclase de PURL, entonces cualquier instancia $\mathcal{F}$ de Ω tal que su fórmula lógicamente equivalente sin literales p-eliminables ($\mathcal{F}^E$) no contenga la cláusula nula es satisfacible.

Para comprender mejor la definición de la clase PURL, merece la pena destacar que ésta surge en un determinado contexto, que es la necesidad de obtener un instrumento para la resolución de problemas geométricos, específicamente el problema de emparejamiento ortogonal en el cilindro, el cual estudiaremos posteriormente con más detalle. La estrategia adoptada para la resolución de este tipo de problemas geométricos pasa por reducir sus entradas al problema de satisfacibilidad y demostrar que, para el algoritmo ELIMINALITERALES($\mathcal{F}$), la familia de fórmulas proposicionales, que representa a las entradas del problema geométrico, posee la propiedad referida en la definición, constituyendo, por tanto, una subclase de PURL.

La clase de fórmulas proposicionales UC-4P-SAT, asociada al problema de etiquetado de puntos alineados UC-4P, constituye un ejemplo de una subclase de PURL (v. Capítulo 3). Y la clase de fórmulas EOSC-SAT, asociada al problema de EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL CILINDRO, una subclase de 2PURL, clase que será definida posteriormente en el Capítulo 4.

Si comparamos la clase PURL, ahora propuesta, con las otras clases de satisfacibilidad polinomiales bien conocidas concluiremos más adelante en este mismo capítulo que cada una de ellas constituye una sub-

clase de PURL. Por lo tanto, esta nueva clase es una extensión de las clases de satisfacibilidad polinomiales bien conocidas.

Considerando la definición de PURL, cada una de sus instancias puede ser resuelta por un algoritmo polinomial de detección y exclusión de literales p-eliminables, por lo que podemos enunciar el siguiente resultado:

Teorema 2.6. PURL *es una clase de satisfacibilidad polinomial.*

Demostración. Sea $\mathcal{F}$ una instancia de PURL. Entonces, por definición, $\mathcal{F}$ es una instancia de una subclase Ω de PURL (v. Definición 2.3). Consideremos en seguida la fórmula proposicional $\mathcal{F}^E$, lógicamente equivalente a $\mathcal{F}$, devuelta por el algoritmo polinomial ELIMINALITERALES($\mathcal{F}$) (v. Lema 2.4). Como $\mathcal{F} \in \Omega$, si $\mathcal{F}$ no es satisfacible, entonces $\mathcal{F}^E$ contiene la cláusula nula, independientemente de la ordenación de la ordenación de sus cláusulas e literales (v. Definición 2.3). Por lo tanto, la satisfacibilidad de $\mathcal{F}$ es determinada en tiempo polinomial. □

2.2. PURL y las clases de satisfacibilidad polinomiales bien conocidas

En la sección anterior definimos la nueva clase de satisfacibilidad PURL, cuyas instancias pueden ser resueltas en tiempo polinomial. Al tratarse de una clase nueva, es importante estudiar su relación con otras clases polinomiales conocidas. A este efecto, consideraremos las clases que Franco y van Gelder (2003) designaron como bien conocidas, las cuales están mostradas en la Figura 1.17. Recordemos que, conforme a lo indicado en esta figura, SLUR y LinAut constituyen dos clases incomparables que, en conjunto, contienen a todas las demás [FG03, maa00].

En la presente sección mostraremos que tanto SLUR como LinAut son subclases de PURL. Como consecuencia de este hecho, y atendiendo a que estas dos clases contienen todas a las demás, podemos concluir que PURL constituye una extensión que contiene a todas a las clases polinomiales bien conocidas (v. Figura 2.1), planteándose de modo natural la cuestión de saber si corresponde a la superclase polinomial conjeturada por Gu, Purdom, Franco y Wah [GPF97].

Como resultado auxiliar y debido al interés que posee la demostración del resultado, probaremos que 2SAT es una subclase de PURL antes

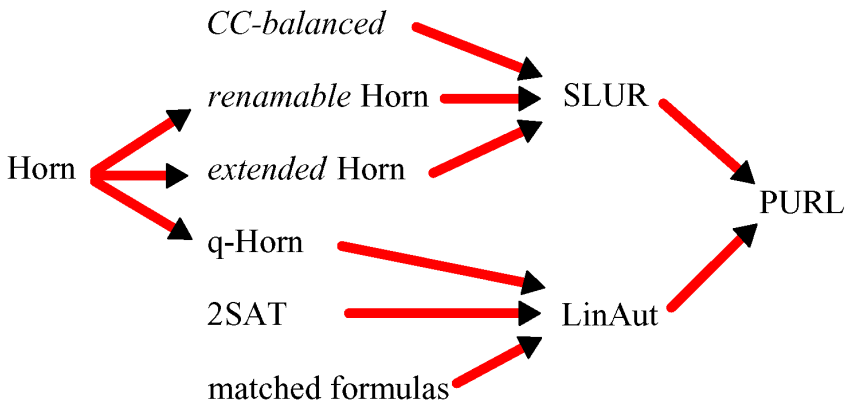


Figura 2.1: Clases polinomiales bien conocidas y su relación de inclusión con PURL.

de comprobar que tanto SLUR como LinAut son subclases propias de PURL.

2SAT es subclase de PURL

Sea $\mathcal{F}$ una instancia 2SAT y $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$ una fórmula lógicamente equivalente sin literales p-eliminables. Si $\square \in \mathcal{F}^E$, sabemos que $\mathcal{F}$ no es satisfacible. Recíprocamente, si suponemos que $\mathcal{F}$ no es satisfacible, mostraremos a continuación que $\square \in \mathcal{F}^E$, lo que nos permite concluir, de acuerdo a la Definición 2.3, que 2SAT es una subclase de PURL.

Para empezar, obtengamos el siguiente resultado previo:

Lema 2.7. *Sean $\mathcal{F}$ una fórmula proposicional* 2SAT *y C una suya cláusula. Para cada literal $l \in C$, la propagación unitaria $(\mathcal{F}^1, \tau) = \text{PROPUNIT}(\mathcal{F} \cup \{[l]\})$ devuelve una fórmula proposicional que contiene la cláusula nula, $\square \in \mathcal{F}^1$ o una subfórmula propia, $\mathcal{F}^1 \subsetneq \mathcal{F}$.*

Demostración. Una cláusula con dos literales al ser modificada se convierte necesariamente o en una cláusula unitaria o en una cláusula nula. Como la fórmula devuelta por el algoritmo de propagación unitaria

no contiene cláusulas unitarias y la cláusula C no pertenece a $\mathcal{F}^1$, se da una de las dos condiciones, $\square \in \mathcal{F}^1$ o $\mathcal{F}^1 \subsetneq \mathcal{F}$. $\square$

En las condiciones del resultado anterior, si $\square \notin \mathcal{F}^1$, las fórmulas proposicionales $\mathcal{F}$ y $\mathcal{F}^1$ son equisatisfacibles, i.e. la condición $\mathcal{F} \equiv \mathcal{F}^1$ es una tautología.

Considerando una instancia 2SAT no satisfacible $\mathcal{F}$, existe una subfórmula, $\mathcal{F}^0 \subseteq \mathcal{F}$, no satisfacible minimal respecto al número de cláusulas. Por lo tanto, para cualquiera de sus cláusulas, $C \in \mathcal{F}^0$ (no nula), todos sus literales son p-eliminables. Siendo $(\mathcal{F}^1, \tau) = \text{PROPUNIT}(\mathcal{F}^0 \cup \mathcal{U}(\overline{Flip(C,l)}))$, se tiene por el resultado anterior (v. Lema 2.7) que $\square \in \mathcal{F}^1$. En el caso contrario, $\mathcal{F}^1 \subsetneq \mathcal{F}^0$ sería satisfacible, pues cualquier subfórmula propia de $\mathcal{F}^0$ es satisfacible, por hipótesis. Excluyendo el literal $l \in C$, $C - [l]$ es una cláusula unitaria, por lo que la propagación unitaria de la fórmula proposicional obtenida substituyendo la claúsula C por la cláusula $C - [l]$ no sería satisfacible. Por lo tanto, el único literal $C - [l]$ es también p-eliminable. Se deduce de esto el siguiente teorema:

Teorema 2.8. 2SAT *es una subclase de* PURL.

Sea $\mathcal{F}$ una instancia SAT. Considerando la fórmula lógicamente equivalente $\mathcal{F}^E$ sin literales p-eliminables devuelta por el algoritmo ELIMINALITERALES, sabemos que si esta fórmula contiene la cláusula nula, la fórmula original no es satisfacible. Por otro lado, según el Teorema 2.8, una instancia 2SAT $\mathcal{F}^E$ sin literales p-eliminables es satisfacible. Este hecho permite definir la siguiente clase de satisfacibilidad proposicional Φ_2, subclase de PURL.

Definición 2.4. Sea $\mathcal{F}$ una fórmula proposicional y $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$. Si, para cualquiera ordenación de las cláusulas y literales de $\mathcal{F}$, $\mathcal{F}^E$ es

1. una instancia 2SAT o
2. una instancia kSAT para la que existe una cláusula C y un literal $l \in C$ tal que $\text{PROPUNIT}(\mathcal{F}^E \cup \mathcal{U}(\overline{Flip(C,l)}))$ es una instancia 2SAT,

diremos que $\mathcal{F}$ es una instancia de la clase de satisfacibilidad proposicional Φ_2.

Nótese que la fórmula proposicional $\{\square\}$ es una instancia 2SAT pues el número de literales en cada cláusula es inferior a 3.

Lema 2.9. *La clase de satisfacibilidad* Φ_2 *es una subclase de* PURL.

Demostración. Sea $\mathcal{F}$ una instancia de Φ_2. Entonces, para cualquiera ordenación de cláusulas y literales de $\mathcal{F}$, la fórmula $\mathcal{F}^E =$

ELIMINALITERALES($\mathcal{F}$) es una instancia en las condiciones (1) o (2) de la Definición 2.4.

$\mathcal{F}$ no es satisfacible si y sólo si $\mathcal{F}^E$ contiene la cláusula nula. Si no fuera así $\mathcal{F}^E$ contendría una subfórmula 2SAT no satisfacible, sin literales p-eliminables y sin la cláusula nula. Lo que contradiría el Teorema 2.8 □

Según la Definición 2.4, la fórmula proposicional (2.1) es una instancia de Φ_2, y por lo tanto de PURL (v. la correspondiente fórmula proposicional sin literales p-eliminables (2.2), que en este caso no depende de la ordenación de sus cláusulas y literales).

§ 2.2.1. SLUR es subclase de PURL.— Recordemos que una fórmula proposicional pertenece a la clase SLUR si para cualquiera secuencia de variables el algoritmo SLUR (v. Algoritmo 2) no finaliza con el mensaje *desisto*.

Consideremos una instancia $\mathcal{F}$ de SLUR. $\mathcal{F}$ no es satisfacible si y sólo si en la línea 1 del algoritmo SLUR (v. Algoritmo 2) la fórmula $(\mathcal{F}^1, \tau) = \text{PROPUNIT}(\mathcal{F})$ contiene la cláusula nula. O sea, la entrada contiene una cláusula unitaria cuyo único literal es p-eliminable. Por tanto, $\mathcal{F}$ no es satisfacible si y sólo si $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$ contiene la cláusula nula, por lo que:

Teorema 2.10. SLUR *es una subclase propia de la clase polinomial* PURL.

Para completar el resultado anterior, probaremos que la clase SLUR está propiamente contenida en PURL. Para ello consideremos consideremos la siguiente fórmula proposicional 3SAT:

$$\begin{aligned}\mathcal{F} = \quad & \{[1,2,-4],[-1,2,4],[2,-3,-4],[-2,3,-4],\\ & [1,2,6],[-1,2,-6],[2,5,-6],[-2,-5,-6],\\ & [1,2,8],[-1,2,-8],[2,-7,-8],[-2,7,-8],\\ & [1,2,-10],[-1,2,10],[2,-9,-10],[-2,9,-10],\\ & [3,4,6],[-3,4,-6],[4,-5,-6],[-4,5,-6],\\ & [3,4,8],[-3,4,-8],[4,-7,-8],[-4,7,-8],\\ & [3,4,10],[-3,4,-10],[4,9,-10],[-4,-9,-10],\\ & [5,6,-8],[-5,6,8],[6,7,-8],[-6,-7,-8],\\ & [5,6,-10],[-5,6,10],[6,-9,-10],[-6,9,-10],\\ & [7,8,10],[-7,8,-10],[8,-9,-10],[-8,9,-10]\}\end{aligned} \tag{2.5}$$

.

Obsérvese que en la línea 8 del algoritmo SLUR (v. Algoritmo 2) se efectúa una elección arbitraria para cada variable v, tomando la fórmula proposicional resuelta con cada uno de los dos literales v y $\overline{v}$. Obsérvese también que una fórmula proposicional pertenece a la clase de satisfacibilidad polinomial SLUR si para todas las posibles secuencias de variables, el algoritmo no devuelve *desisto* como respuesta.

Si comenzamos por la variable v_{10} en la fórmula (2.5), representada por su respectivo índice, la propagación unitaria de cada una de las atribuciones $\{10\}$ y $\{-10\}$ no produce la cláusula nula. Sin embargo, en la elección arbitraria (v. línea 8 del Algoritmo 2) tomando la fórmula proposicional obtenida por propagación unitaria del literal afirmado, el algoritmo finaliza con el mensaje *desisto*. Por lo tanto, esta fórmula proposicional no es una instancia de SLUR.

Excluyendo los literales p-eliminables en la fórmula (2.5) con el algoritmo ELIMINALITERIAS se obtiene la siguiente fórmula proposicional:

$$\mathcal{F}^E = \{[2], [3], [-6], [-5], [-8], [7], [-10]\}. \tag{2.6}$$

Resultado que no depende de la ordenación de las cláusulas y de los literales de $\mathcal{F}$ porque cada cláusula $C \in \mathcal{F}$ tiene un literal p-eliminable (en $\mathcal{F}$), como mínimo.

Por lo que $\mathcal{F}$ es una instancia de Φ_2 y, por consiguiente, de PURL. Esto demuestra que la clase polinomial SLUR es una subclase propia de PURL.

§ 2.2.2. LinAut es subclase de PURL.— De forma similar a como hemos hecho anteriormente, mostraremos que una instancia de LinAut no es satisfacible si y solo si la fórmula derivada sin literales p-eliminables contiene la cláusula nula. Por lo tanto es suficiente mostrar que si $\mathcal{F}$ no es satisfacible entonces $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$ contiene la cláusula nula.

Si consideramos $\mathcal{F}$ una fórmula proposicional perteneciente a la clase LinAut, existe una *autarquía lineal* $\mathrm{x} \in Q^n$ y una atribución parcial τ asociada de modo que, si $C \in \mathcal{F}$ es una cláusula conteniendo alguna variable atribuida entonces $C(\tau) = 1$. En estas condiciones, denotando por $\mathcal{F}^{'} = \{C \in \mathcal{F} : C(\tau_i) = 1\}$, $\mathcal{F}^{'} \subset \mathcal{F}$ y $\mathcal{F} - \mathcal{F}^{'}$ es también una instancia de LinAut.

Denotemos $\mathcal{F}_1 = \mathcal{F} - \mathcal{F}^{'}$. La subfórmula $\mathcal{F}_1 \subset \mathcal{F}$ es también una instancia de LinAut, por lo que existe una atribución de verdad satisfaciendo todas las cláusulas de $\mathcal{F}^{''} \subset \mathcal{F}_1$ sin que ninguna de las cláusulas de $\mathcal{F}_1 - \mathcal{F}^{''}$ sea afectada por la atribución de verdad, i.e., las cláusulas de $\mathcal{F}_1 - \mathcal{F}^{''}$ no contienen ningún literal de la referida atribución ni ninguno de sus complementarios. Denotemos $\mathcal{F}_2 = \mathcal{F}_1 - \mathcal{F}^{''}$. $\mathcal{F}_2$ está contenido en $\mathcal{F}_1$ y es una instancia LinAut. Repitiendo la construcción un número

finito de veces (puesto que el número de cláusulas y literales es finito) se obtiene una secuencia de subfórmulas

$$\mathcal{F}_k \subset \mathcal{F}_{k-1} \subset \cdots \subset \mathcal{F}_1 \subset \mathcal{F}.$$

La fórmula $\mathcal{F}_k$ es una fórmula vacía o una instancia 2SAT no satisfacible. En el primer caso $\mathcal{F}$ es una fórmula satisfacible y, en el segundo, una fórmula no satisfacible.

Recuérdese que, dada una fórmula proposicional $\mathcal{F}$ y la correspondiente matriz cláusula-variable, M, ésta admite una *autarquía lineal* $\mathbf{x} \in \mathbb{Q}^n$ si

$$\begin{cases} M\mathbf{x} \geq \mathbf{0} \\ \mathbf{x} \neq \mathbf{0} \end{cases} \tag{2.7}$$

Cuando existe una *autarquía lineal*, $\mathbf{x} \in \mathbb{Q}^n$, existe también una *asignación de autarquía*, τ, de modo que $v_j \in \tau$ si $sgn(x_j) = 1$ y $\overline{v_j} \in \tau$ se $sgn(x_j) = -1$. Una autarquía lineal puede determinarse en tiempo polinomial utilizando un algoritmo de programación lineal [kul00].

Teorema 2.11. *LinAut es una subclase propia de la clase de satisfacibilidad polinomial* PURL.

Demostración. Sea $\mathcal{F}$ una instancia de LinAut no satisfacible. Por la observación anterior, esta instancia contiene una subfórmula 2SAT no satisfacible por lo que la fórmula proposicional lógicamente equivalente sin literales p-eliminables, $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$ contiene la cláusula nula. Recíprocamente, si $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F})$ contiene la cláusula nula, $\mathcal{F}$ no es satisfacible. Por lo tanto una instancia LinAut no es satisfacible si y solo si la fórmula equivalente sin literales p-eliminables contiene la cláusula nula. En estas condiciones, de acuerdo con la Definición 2.3, LinAut es subclase de PURL. Para demostrar que las dos clases son distintas y que, por tanto, la inclusión es estricta véase el ejemplo a continuación. □

Consideremos nuevamente la fórmula proposicional $\mathcal{F}$ en (2.5):

$$\begin{aligned} \mathcal{F} = \quad & \{[1,2,-4],[-1,2,4],[2,-3,-4],[-2,3,-4], \\ & [1,2,6],[-1,2,-6],[2,5,-6],[-2,-5,-6], \\ & [1,2,8],[-1,2,-8],[2,-7,-8],[-2,7,-8], \\ & [1,2,-10],[-1,2,10],[2,-9,-10],[-2,9,-10], \\ & [3,4,6],[-3,4,-6],[4,-5,-6],[-4,5,-6], \\ & [3,4,8],[-3,4,-8],[4,-7,-8],[-4,7,-8], \\ & [3,4,10],[-3,4,-10],[4,9,-10],[-4,-9,-10], \\ & [5,6,-8],[-5,6,8],[6,7,-8],[-6,-7,-8], \\ & [5,6,-10],[-5,6,10],[6,-9,-10],[-6,9,-10], \\ & [7,8,10],[-7,8,-10],[8,-9,-10],[-8,9,-10]\} \end{aligned}$$

.

Por definición, es condición necesaria para que una fórmula proposicional sea una instancia de LinAut que la correspondiente matriz cláusula-variable satisfaga la ecuación (2.7), i.e, que exista $\mathbf{x} \neq \mathbf{0} \in \mathbb{Q}^n$ tal que $M\mathbf{x} \geq \mathbf{0}$. Sin embargo, para la fórmula proposicional $\mathcal{F}$ y su correspondiente matriz cláusula-variable M (2.8), la ecuación $M\mathbf{x} \geq \mathbf{0}$ tiene solución única con todos los valores de x_i nulos, por lo que $\mathcal{F}$ no es una instancia de la clase polinomial LinAut.

$$M = \begin{bmatrix}
1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1
\end{bmatrix} \tag{2.8}$$

Para probar esto, basta combinar las inecuaciones correspondientes a las líneas 3 y 4 de la matriz M, de lo que resulta que $-x_4 \geq 0$. De la combinación de las líneas 17 y 18, se tiene $x_4 \geq 0$. Por tanto, necesariamente, $x_4 = 0$. Análogamente, de las líneas 7, 8, 29 y 30, $x_6 = 0$. Considerando las líneas 11, 12, y 38, se obtiene $x_8 = 0$. Con las líneas $17 - 18$, $19 - 20$, y $23 - 24$ resulta que $x_3 = x_5 = x_7 = 0$. Con las inecuaciones 33 y 34 se deduce que $x_{10} = 0$ y de las 39 y 40, $x_9 = 0$. Usando las

líneas 11 y 12, $x_2 = 0$ y, finalmente, por las inecuaciones 1 y 2, $x_1 = 0$.

Se concluye por tanto que el sistema (2.7) no tiene solución no trivial (solución no nula), como habíamos previsto.

Considerando el último par de resultados se puede enunciar el siguiente teorema:

Teorema 2.12. PURL *es una superclase para las clases de satisfacibilidad polinomiales* bien conocidas.

Demostración. Las clases polinomiales SLUR y LinAut son distintas pero, en conjunto, contienen todas las demás clases polinomiales *bien conocidas* [maa00, FG03] (v. Figura 1.17). Como, de acuerdo con los teoremas 2.10 y 2.11, cada una de estas es subclase de PURL, por lo tanto cada una de las clases polinomiales *bien conocidas* es subclase de PURL. □

2.3. Buscando modelos en instancias PURL. (El algoritmo SOLVELIMLIT.)

Una de las preguntas que debemos hacernos respecto a una clase de satisfacibilidad es el reconocimiento de sus instancias. Existen clases para las cuales esta cuestión tiene respuesta simple, e.g. 2SAT y Horn. En otros casos, la respuestas es menos obvia, pero el reconocimiento puede ser realizado mediante algoritmos polinomiales, e.g. q-Horn y LinAut. Finalmente, para otras clases no se conocen algoritmos que, en tiempo polinomial, permitan decidir si una fórmula proposicional es o no una instancia de una clase determinada de satisfacibilidad polinomial, e.g. *extended* Horn y SLUR.

Aún no se conoce ningún algoritmo polinomial que permita determinar si una instancia arbitraria del problema de satisfacibilidad es o no una instancia de PURL. Este aspecto merecerá desarrollar una línea de investigación en el futuro, pues este reconocimiento es importante para la resolución del problema de satisfacibilidad de sus instancias cuando nos basamos en los algoritmos de detección y exclusión de literales p-eliminables.

Como fue dicho anteriormente, si el algoritmo ELIMINALITERALES($\mathcal{F}$) devuelve una fórmula proposicional conteniendo la cláusula nula, sabemos que no es satisfacible. Si la fórmula lógicamente equivalente sin literales p-eliminables obtenida no contiene la cláusula nula, no puede deducirse nada a menos que se sepa que es una instancia de una de las subclases de PURL conocidas, como 2SAT [IA86], Φ_2, LinAut [kul00],

SLUR [FG03] o UC-4P-SAT. Esta última será estudiada extensivamente en el Capítulo 3. Sin embargo, la exclusión de literales p-eliminables sucede cuando localmente existen *bloqueos* a la satisfacibilidad. Este aspecto nos motiva a desarrollar un algoritmo que, a partir de una fórmula lógicamente equivalente sin literales p-eliminables que no contenga la cláusula nula, haga la búsqueda de la existencia de un modelo que satisfaga la referida fórmula y consecuentemente la fórmula inicial.

Con este objetivo se ha desarrollado el algoritmo SOLVELIMLIT, basado en el algoritmo ELIMINALITERALES. Tomando como entrada una fórmula proposicional, SOLVELIMLIT empieza por determinar la correspondiente fórmula sin literales p-eliminables. A continuación atribuye, en cada paso, el valor verdadero a un literal de una de sus cláusulas y propaga la atribución a la fórmula. Antes de pasar al siguiente literal, se excluyen todos los literales p-eliminables de la fórmula obtenida, eliminando así los bloqueos locales que pudieran existir. El algoritmo termina cuando no existan más cláusulas o cuando, en algún paso, el algoritmo ELIMINALITERALES($\mathcal{F}$) produzca una cláusula nula. En el primer caso podemos concluir que la fórmula es satisfacible y que la atribución de verdad obtenida es un modelo para la fórmula. En el segundo caso, nada se puede concluir acerca de la satisfacibilidad de la fórmula proposicional por lo que el algoritmo devuelve el mensaje *desisto* (v. Algoritmo 4).

Algoritmo 4 SOLVELIMLIT($\mathcal{F}$)

Entrada: Una fórmula proposicional $\mathcal{F}$
Salida: Un modelo τ satisfaciendo $\mathcal{F}$ o el mensaje *no satisfacible* o *desisto*

```
    F^E = ELIMINALITERALES(F)
    if □ ∈ F^E then
      return no satisfacible
    end if
 5: (F, τ)=PROPUNIT(F^E)
    while F ≠ ∅ do
      para una cláusula, escoger un literal l ∈ C
      F' = ELIMINALITERALES(F ∪ {[l]})
      if □ ∈ F' then
10:     return desisto
      end if
      (F'', τ')=PROPUNIT(F')
      (F, τ) = (F'', τ ∪ τ')
    end while
15: return τ
```

El algoritmo ELIMINALITERALES($\mathcal{F}$) corre en tiempo polinomial, de

hecho $O(|\mathcal{F}|^3)$ en el peor caso. Como el algoritmo es ejecutado a lo más una vez por cada cláusula de $\mathcal{F}$, resulta que también el algoritmo SOLVELIMLIT($\mathcal{F}$) es polinomial con complejidad $O(c|\mathcal{F}|^3)$ en el peor caso, dónde $c = min\{m, n\}$ siendo n el número de variables y m el número de cláusulas de la fórmula proposicional.

A semejanza del algoritmo SLUR, SOLVELIMLIT puede ser utilizado con cualquier instancia de SAT, sea PURL o no.

Lema 2.13. *Si $\mathcal{F}$ es una instancia de* SLUR*, el algoritmo* SOLVELIMLIT($\mathcal{F}$) *nunca devuelve* desisto *como repuesta.*

Demostración. Sea $\mathcal{F}^E$ la fórmula proposicional lógicamente equivalente a $\mathcal{F}$ obtenida en la línea 1 del algoritmo SOLVELIMLIT($\mathcal{F}$) (v. Algoritmo 4). Si $\mathcal{F}$ no es satisfacible, se tiene que por hipótesis esta fórmula es una instancia de SLUR y el algoritmo de propagación unitaria retorna una fórmula con la cláusula nula (v. línea 1 del Algoritmo 2, SLUR). Por lo tanto se tiene que $\square \in \mathcal{F}^E$ y, por consiguiente, SOLVELIMLIT($\mathcal{F}$) termina con el mensaje *no satisfacible*.

Supongamos ahora que $\mathcal{F}$ es satisfacible. En el caso de que $\mathcal{F}^E$ contenga alguna cláusula unitaria, la atribución obtenida en la línea 5 del Algoritmo 4, τ^0, contiene algún literal. Sea $\tau^0 = \{l_1, l_2, \cdots, l_{r_0}\}$ esta atribución de verdad y sea $V_0 = \{v_1, \cdots, v_{r_0}\}$ la secuencia de variables tal que l_i es uno de los dos literales de la variable v_i, para cada $l_i \in \tau^0$. Considerando el algoritmo SLUR, para la variable v_1 en la rama correspondiente al literal l_1, $(\mathcal{F}^1, \tau^1) = \text{PROPUNIT}(\mathcal{F} \cup \{[l_1]\})$, la fórmula $\mathcal{F}^1$ no contiene la cláusula nula por lo es posible escoger esta atribución. Así sucesivamente para cada una de las variables del conjunto V_0.

Consideremos el literal $l \in C$ elegido en la línea 7 del algoritmo SOLVELIMLIT y sea v su correspondiente variable lógica. En el algoritmo SLUR, en ninguno de los casos $(\mathcal{F}^1, \tau^1) = \text{PROPUNIT}(\mathcal{F} \cup \{[l]\})$ y $(\mathcal{F}^2, \tau^2) = \text{PROPUNIT}(\mathcal{F} \cup \{[\bar{l}]\})$, la fórmula obtenida contiene a la cláusula nula, pues si no fuera así o bien $l \in C$ es p-eliminable (y en este caso no lo hubiéramos podido elegir) o cada uno de los literales de $C - [l]$ sería p-eliminable (por lo que $|C| = 1$), en contradicción con el hecho de que tras la instrucción de la línea 5, la fórmula no contiene cláusulas unitarias.

Supongamos, por reducción al absurdo, que en la línea 9 del algoritmo SOLVELIMLIT $\square \in \mathcal{F}'$ y que, por tanto, este algoritmo termina con el mensaje *desisto*. Entonces, ninguno de los modelos de $\mathcal{F}$ contiene a la atribución parcial $\tau^0 \cup \{l\}$. Pero, en estas condiciones, el algoritmo SLUR termina con el mensaje *desisto*, en contradicción con la hipótesis de partida. Concluimos, por tanto, que el algoritmo SOLVELIMLIT nunca finaliza con el mensaje *desisto*. □

Tras establecer esta relación entre las clases SLUR y SOLVELIMLIT,

se ha desarrollado un estudio comparativo entre ambos algoritmos y su capacidad de resolución de fórmulas proposicionales sin previo reconocimiento. Para ello se ha utilizado una colección de conjuntos test, la cual se usa habitualmente para el análisis comparativo de algoritmos de satisfacibilidad[1]. Los resultados obtenidos están sintetizados en el Cuadro 2.1 y serán presentados en la Sección 2.5.

Al contrario de SLUR, cuyas instancias se resuelven usando el algoritmo SOLVELIMLIT, hay fórmulas LinAut que no pueden resolverse mediante por este algoritmo (SOLVELIMLIT). Consideremos la fórmula proposicional no satisfacible *dubois20.cnf*[1] (v. Ecuación 2.9). Aunque no es satisfacible, esta fórmula, $\mathcal{F}$, no contiene literales p-eliminables. Consideremos ahora dos nuevas variables 61 y 62, añadamos el literal afirmado 61 a cada una de las cláusulas y añadamos una nueva cláusula $[-61, 62]$ se obtiene una nueva fórmula, $\mathcal{F}^1$, que es una instancia de LinAut satisfacible y que no tiene literales p-eliminables. En el algoritmo SOLVELIMLIT, si en la línea 7 se atribuye el valor 1 al literal negado -61, el algoritmo finaliza con el mensaje *desisto*.

$$
\begin{aligned}
\mathcal{F} = \quad & [39, 40, 1], [-39, -40, 1], [39 - 40, -1], [-39, 40, -1], [1, 41, 2], \\
& [-1, -41, 2], [1, -41, -2], [-1, 41, -2], [2, 42, 3], [-2, -42, 3], \\
& [2, -42, -3], [-2, 42, -3], [3, 43, 4], [-3, -43, 4], [3, -43, -4], \\
& [-3, 43, -4], [4, 44, 5], [-4, -44, 5], [4, -44, -5], [-4, 44, -5], \\
& [5, 45, 6], [-5, -45, 6], [5, -45, -6], [-5, 45, -6], [6, 46, 7], \\
& [-6, -46, 7], [6, -46, -7], [-6, 46, -7], [7, 47, 8], [-7, -47, 8], \\
& [7, -47, -8], [-7, 47, -8], [8, 48, 9], [-8 - 48, 9], [8 - 48, -9], \\
& [-8, 48, -9], [9, 49, 10], [-9, -49, 10], [9, -49, -10], [-9, 49, -10], \\
& [10, 50, 11], [-10, -50, 11], [10, -50, -11], [-10, 50, -11], [11, 51, 12], \\
& [-11, -51, 12], [11, -51, -12], [-11, 51, -12], [12, 52, 13], [-12, -52, 13], \\
& [12, -52, -13], [-12, 52, -13], [13, 53, 14], [-13, -53, 14], [13, -53, -14], \\
& [-13, 53, -14], [14, 54, 15], [14, -54, 15], [14, -54, -15], [-14, 54, -15], \\
& [15, 55, 16], [-15, -55, 16], [15, -55, -16], [-15, 55, -16], [16, 56, 17], \\
& [-16, -56, 17], [16, -56, -17], [-16, 56, -17], [17, 57, 18], [-17, -57, 18], \\
& [17, -57, -18], [-17, 57, -18], [18, 58, 19], [-18, -58, 19], [18, -58, -19], \\
& [-18, 58, -19], [19, 59, 60], [-19, -59, 60], [19, -59, -60], [-19, 59, -60], \\
& [20, 59, 60], [-20, -59, 60], [20, -59, -60], [-20, 59, -60], [21, 58, 20], \\
& [-21, -58, 20], [21, -58, -20], [-21, 58, -20], [22, 57, 21], [-22, -57, 21], \\
& [22, -57, -21], [-22, 57, -21], [23, 56, 22], [-23, -56, 22], [23, -56, -22], \\
& [-23, 56, -22], [24, 55, 23], [-24, -55, 23], [24, -55, -23], [-24, 55, -23], \\
& [25, 54, 24], [-25, -54, 24], [25, -54, -24], [-25, 54, -24], [26, 53, 25], \\
& [-26, -53, 25], [26, -53, -25], [-26, 53, -25], [27, 52, 26], [-27, -52, 26], \\
& [27, -52, -26], [-27, 52, -26], [28, 51, 27], [-28, -51, 27], [28, -51, -27], \\
& [-28, 51, -27], [29, 50, 28], [-29, -50, 28], [29, -50, -28], [-29, 50, -28], \\
& [30, 49, 29], [-30, -49, 29], [30, -49, -29], [-30, 49, -29], [31, 48, 30], \\
& [-31, -48, 30], [31, -48, -30], [-31, 48, -30], [32, 47, 31], [-32, -47, 31], \\
& [32, -47, -31], [-32, 47, -31], [33, 46, 32], [-33, -46, 32], [33, -46, -32], \\
& [-33, 46, -32], [34, 45, 33], [-34, -45, 33], [34, -45, -33], [-34, 45, -33], \\
& [35, 44, 34], [-35, -44, 34], [35, -44, -34], [-35, 44, -34], [36, 43, 35], \\
& [-36, -43, 35], [36, -43, -35], [-36, 43, -35], [37, 42, 36], [-37, -42, 36], \\
& [37, -42, -36], [-37, 42, -36], [38, 41, 37], [-38, -41, 37], [38, -41, -37], \\
& [-38, 41, -37], [39, 40, -38], [-39, -40, -38], [39, -40, 38], [-39, 40, 38],
\end{aligned}
\tag{2.9}
$$

[1] `http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`

2.4. Jerarquías de clases PURL

Algunas fórmulas proposicionales que no pertenecen a ninguna de las clases polinomiales conocidas pueden ser reducidas a fórmulas *equivalentes* pertenecientes a alguna de esas clases. Si esta reducción es eficiente entonces estas fórmulas pueden ser resueltas también en tiempo polinomial. Las reducciones se hacen por niveles, correspondiendo cada nivel a una clase de satisfacibilidad polinomial. Generalmente al nivel más bajo le corresponde menor complejidad temporal.

Un ejemplo de jerarquía de clases de satisfacibilidad puede encontrarse en el trabajo de Dalal y Etherington (1992) [DE92]. Estos autores definen una familia de clases de satisfacibilidad, $\Omega_0 \subset \Omega_1 \subset \cdots \subset \Omega_k$, dónde Ω_0 contiene a las clases Horn y 2SAT. Para resolver las instancias de la clase base, Ω_0, en tiempo lineal respecto al tamaño da entrada ($O(\mathcal{F})$), los mismos autores desarrollaron un algoritmo basado en propagación unitaria. De un modo general, una fórmula proposicional $\mathcal{F}$ pertenece a la clase Ω_k si, para alguna variable $v \in var(\mathcal{F})$, la fórmula proposicional obtenida por propagación unitaria de una de las dos posibles atribuciones de v, $\{v\}$ o $\{\overline{v}\}$ es una instancia de la clase del nivel inferior, Ω_{k-1}. De este modo, por recurrencia, una instancia de esta clase (de nivel k) puede ser resuelta en tiempo $O(|\mathcal{F}|n^k)$, dónde n designa el número de variables en la fórmula proposicional, $\mathcal{F}$.

Otros ejemplos de jerarquías de clases polinomiales pueden ser encontrados en los trabajos de Gallo y Scutella (1988) [GS88] y de Pretolani (1996) [pre96].

Anteriormente, en el presente capítulo, definimos una nueva clase de satisfacibilidad que designamos por PURL, basada en los conceptos de literal eliminable y p-eliminable. Recuérdese que un literal l en una cláusula C de una fórmula $\mathcal{F}$ se llama p-eliminable si la propagación unitaria de la atribución de verdad $\overline{Flip(C,l)}$ falsifica $\mathcal{F}$.

Para abreviar, designaremos a partir de ahora $\overline{Flip(C,l)} * \mathcal{F}$ a la resolución (en el sentido de [ZS96]) de la fórmula proposicional $\mathcal{F}$ con la atribución de verdad $\overline{Flip(C,l)}$.

Podemos ampliar el concepto de literal p-eliminable mediante un concepto similar al de las jerarquías que acabamos de citar. Dada una cláusula $C \in \mathcal{F}$, diremos que uno de sus literales l es 2p-eliminable si la fórmula proposicional $\mathcal{F}$ resuelta con la atribución de verdad $\overline{Flip(C,l)}$, $\overline{Flip(C,l)} * \mathcal{F}$, contiene una cláusula con todos los literales p-eliminables.

En estas condiciones, si el literal $l \in C$ es 2p-eliminable, cualquier atribución de verdad conteniendo $\overline{Flip(C,l)}$ falsifica $\mathcal{F}$, por lo que $Flip(C,l)$ es una consecuencia lógica de $\mathcal{F}$. Por el Lema 2.3, el literal $l \in C$ es eliminable. Excluyéndolo (de la cláusula C) se obtiene una fórmula proposicional lógicamente equivalente, $\mathcal{F}'$. Iterando el proceso

hasta que no existan literales de este tipo se obtiene una fórmula proposicional lógicamente equivalente $\mathcal{F}^E$ sin literales 2p-eliminables.

Procediendo de modo análogo, diremos que un literal $l \in C$ es 3p-eliminable si la fórmula proposicional $\overline{Flip(C,l)} * \mathcal{F}$ contiene una cláusula con todos los literales 2p-eliminables.

De un modo general, diremos que un literal $l \in C$ es rp-eliminable, $r > 1$, si la fórmula proposicional $\overline{Flip(C,l)} * \mathcal{F}$ contiene una cláusula con todos sus literales $(r-1)$p-eliminables.

Para identificar y excluir los literales rp-eliminables se ha desarrollado el algoritmo ELIMINALITERALES$(\mathcal{F}, r)$ (v. Algoritmo 5) a partir del algoritmo ELIMINALITERALES$(\mathcal{F})$ (v. Algoritmo 3). Este algoritmo asume como entrada el par $(\mathcal{F}, r)$ formado por una fórmula proposicional y un entero r correspondiente al tipo de literales a excluir. Para $r = 1$ el algoritmo ELIMINALITERALES$(\mathcal{F}, r)$ coincide con el algoritmo ELIMINALITERALES$(\mathcal{F})$.

Para $r = 2$, el algoritmo detecta y excluye literales 2p-eliminables. Dado que en cada paso se ejecuta el algoritmo ELIMINALITERALES$(\mathcal{F})$ y que, en el peor caso, en cada iteración sobre todas las cláusulas y literales de $\mathcal{F}$ es eliminado un único literal, el algoritmo corre en tiempo $O(|\mathcal{F}|^2|\mathcal{F}^3|)$. Por lo tanto, en el caso general, ELIMINALITERALES$(\mathcal{F}, r)$ tiene complejidad $O(|\mathcal{F}|^{2r+1})$.

Caracterizado el concepto de literal rp-eliminable y desarrollado un algoritmo que permite detectar y eliminar este tipo de literales eliminables, la familia de clases de satisfacibilidad rPURL se define del siguiente modo:

Definición 2.5. Sea Ω una clase de fórmulas proposicionales. Ω es una subclase de rPURL si y sólo si para cada fórmula proposicional $\mathcal{F} \in \Omega$ no satisfacible, $\mathcal{F}^E =$ ELIMINALITERALES$(\mathcal{F}, r)$ contiene a la cláusula nula (independientemente de la ordenación de sus cláusulas y literales).

Considerando la definición de literal rp-eliminable y la definición anterior, las clases: PURL, 2PURL, $\cdots$, rPURL, constituyen una jerarquía de clases de satisfacibilidad.

Por otro lado, para cualquier $r > 1$, cualquier subclase, Ω, de $(r-1)$PURL es también subclase de rPURL. En efecto, si $\mathcal{F}$ es una instancia no satisfacible de $\Omega \subset (r-1)$PURL, esto significa que el algoritmo ELIMINALITERALES$(\mathcal{F}, r)$ devuelve una fórmula proposicional con la cláusula nula. Por tanto,

$$\text{PURL} \subset 2\text{PURL} \subset \cdots \subset r\text{PURL}.$$

Similarmente a la clase de satisfacibilidad PURL, cada una de las clases rPURL es definida a partir de sus subclases. El concepto es más

Algoritmo 5 ELIMINALITERALES($\mathcal{F}, r$) (Elimina literales rp-eliminables)

Entrada: Una CNF-fórmula $\mathcal{F}$ y un número entero r
Salida: Una fórmula $\mathcal{F}$ sin literales rp-eliminables

```
if r=1 then
  F = ELIMINALITERALES(F)
  return (F)
else
  repeat
    remlitfree = true
    for all C ∈ F do
      for all l ∈ C do
        F' = ELIMINALITERALES(F ∪ U(Flip(C,l)), r − 1)
        if □ ∈ F' then
          F = (F − {C}) ∪ ({C − [l]}), remlitfree = false
          if □ ∈ F then
            F = {□}
          end if
        end if
      end for
    end for
  until F = {□} OR remilitfree = true
  return (F)
end if
```

que un resultado teórico, pues son conocidas clases de satisfacibilidad asociadas a problemas geométricos que constituyen subclases de esta jerarquía ahora definida. UC-4P-SAT es una subclase de PURL y EOSC-SAT de 2PURL, como veremos más adelante en este trabajo.

De la investigación desarrollada hasta el momento y que continuará en el futuro, se conjetura que el problema de satisfacibilidad asociado al problema de etiquetado de puntos alineados con etiquetas rectangulares en posiciones fijas, OR-4P sea una subclase de PURL, siempre que el ancho de las etiquetas sea igual o superior a su altura. También se conjetura que el caso general sin restricciones corresponde a una subclase de 2PURL.

A partir de las semejanzas encontradas con el problema EOSC, se conjetura que el problema de EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL TORO (EOST) corresponde a una subclase de 3PURL.

Estos ejemplos sugieren que pueden existir muchos más problemas, de naturaleza geométrica o no, reducibles a clases de satisfacibilidad polinomiales rPURL, para un valor de r adecuado.

Existe una generalización evidente del algoritmo SOLVELIMLIT($\mathcal{F}$)

para obtener la resolución de algunas de las instancias de PURL. Designaremos por SOLVELIMLIT$(\mathcal{F}, r)$ al algoritmo que se obtiene a partir del Algoritmo 4 reemplazando la instrucción de la línea 1 por $\mathcal{F}^E =$ ELIMINALITERALES$(\mathcal{F}, r)$.

El algoritmo ELIMINALITERALES$(\mathcal{F}, r)$ se ejecuta en tiempo $O(|\mathcal{F}|^{2r+1})$. Por lo tanto, el algoritmo SOLVELIMLIT$(\mathcal{F}, r)$ corre en tiempo $O(|\mathcal{F}|^{2r+1}+c|\mathcal{F}|^3)$, dónde $c = min\{m, n\}$. Si $r > 1$, SOLVELIMLIT$(\mathcal{F}, r)$ corre en tiempo $O(|\mathcal{F}|^{2r+1})$.

De un modo similar al algoritmo SOLVELIMLIT$(\mathcal{F})$, este algoritmo al terminar devuelve un modelo para $\mathcal{F}$ o uno de los mensajes *no satisfacible* o *desisto*.

2.5. Ejemplos, tests y conclusiones

La presente sección se centra fundamentalmente en el análisis experimental de los algoritmos SOLVELIMLIT$(\mathcal{F})$ y SOLVELIMLIT$(\mathcal{F}, r)$ y de una aplicación de los algoritmos ELIMINALITERALES$(\mathcal{F})$ y ELIMINALITERALES$(\mathcal{F}, r)$ a la resolución de un conjunto de rompecabezas SUDOKU.

Considerando algunas semejanzas existentes entre los algoritmos SLUR$(\mathcal{F})$ y SOLVELIMLIT$(\mathcal{F})$, presentaremos también un análisis comparativo de los resultados obtenidos por cada uno de estos dos algoritmos.

Recuérdese que para la clase SLUR, el problema de reconocimiento no está resuelto y que es, probablemente, un problema intratable. En la nueva clase PURL sucede algo semejante. Recordemos que se puede ejecutar el algoritmo SLUR con una instancia arbitraria. En este caso, este algoritmo puede finalizar con un modelo, con el mensaje *no satisfacible* o con el mensaje *desisto*.

Según Franco y Gelder, de las dos clases de satisfacibilidad polinomiales SLUR y LinAut, incomparables entre si y que en conjunto contienen a todas las demás clases polinomiales bien conocidas, LinAut es predominante [FG03].

Sea $\mathcal{F}$ una variable aleatoria en el espacio de las kCNF fórmulas proposicionales formadas por m cláusulas, con k literales cada una, tomadas uniforme e independientemente sobre n variables booleanas. Para el estudio de la medida, consideremos la distribución de probabilidad $\mathcal{M}^k_{m,n}$ [FG03]. Para la definición de la distribución $\mathcal{M}^k_{m,n}$ obsérvese que el espacio de las cláusulas está constituido por $2^k C^n_k$ cláusulas con k literales cada una, que no contienen simultáneamente a un literal y su

complementario. El espacio de las fórmulas (sucesos) está formado por todas las secuencias de m cláusulas, cada una con igual probabilidad, $\left(2^k C_k^n\right)^{-m}$.

Los valores del espacio de probabilidad están muy asociados a los valores del parámetro $r = \frac{m}{n}$. Por ello, se estudiará el comportamiento asintótico cuando m y n crecen, mientras k se mantiene fijo.

Conforme a los resultados teóricos y a las observaciones experimentales, el cociente $r = \frac{m}{n}$ proporciona un indicador de *dureza* o dificultad de la resolución de las fórmulas proposicionales aleatorias formando un patrón *fácil, difícil, fácil*. Considerando el espacio de las kCNF fórmulas aleatorias con distribución de probabilidad $\mathcal{M}_{m,n}^k$ y $k \geq 3$ se tienen las siguientes propiedades:

1. Para $r \geq 2^k ln(2)$, hay una probabilidad muy elevada de que una fórmula no sea satisfacible, i.e. la probabilidad de que $\mathcal{F}$ sea una fórmula no satisfacible tiende a 1 cuando m y n tienden a ∞.

2. Para cualquier valor fijo $r < \frac{2^k}{4k}$, una fórmula proposicional es satisfacible con elevada probabilidad [CM92, CF86]. En el caso particular de que $k = 3$, la cota $r = 3.003$ fue establecida por Frieze y Suen [FS96].

3. Para un valor fijo $r < \dfrac{2^k}{4k}$, una fórmula satisfacible puede ser resuelta en tiempo lineal por un algoritmo que elimine variables iterativamente, comenzando por las cláusulas más pequeñas [FS96, CM92, CF86].

4. Para cualquier $r > \frac{2}{k(k-1)}$ una fórmula aleatoria tiene una probabilidad elevada de no pertenecer ni a la clase SLUR ni a la clase q-Horn [FG03].

5. Para cualquier valor $r < 0.64$ una fórmula aleatoria pertenece a la clase *matched fórmulas* con una probabilidad elevada. Ésta aumenta con k pues para $k = 4$, $r < 0.84$, aproximándose a 1 cuando k crece.

Sin embargo, la existencia de emparejamientos (permitiendo resolver el problema de satisfacibilidad de las *matched fórmulas*) sólo es posible cuando $m \leq n$. Recordemos que la clase *matched fórmulas* está propiamente contenida en LinAut. Estos hechos fueron los que motivaron a Franco y Gelder a concluir que, entre las clases SLUR y LinAut, esta última es predominante.

Observamos anteriormente que el algoritmo SOLVELIMLIT resuelve la satisfacibilidad de todas las instancias de SLUR. Así, de un modo

general, es de esperar que para las instancias con las que el algoritmo SLUR no *desista*, el algoritmo SOLVELIMLIT($\mathcal{F}$) tampoco termine con el mensaje *desisto*. Esto es lo que ha ocurrido en todas las pruebas realizadas, que se presentarán luego en esta misma sección.

También se ha observado que el algoritmo SOLVELIMLIT no resuelve el problema de satisfacibilidad de todas las fórmulas proposicionales en LinAut (apenas se puede garantizar por las no satisfacibles), por lo que este algoritmo no resuelve la satisfacibilidad de las instancias de PURL. SOLVELIMLIT no es un resolutor de fórmulas PURL, sino más bien un intento de mostrar, de cierto modo, el potencial de uso del concepto de literal p-eliminable.

En el presente estudio comparativo se considerarán diversos conjuntos de fórmulas proposicionales habitualmente utilizados para examinar la efectividad de algoritmos de satisfacibilidad, disponibles en la dirección: `http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`.

La descripción y caracterización de las instancias de cada uno de los conjuntos utilizados se encuentra en el mismo lugar.

Para cada uno de los conjuntos considerados, el Cuadro 2.1 muestra la identificación del conjunto (*aim*, *bms*, *cbs*, etc), el número de instancias del mismo, la indicación del número de instancias cuya respuesta fue *satisfacible* (#Sat), *no satisfacible* (#Unsat) y *desisto* (#Desisto), para cada uno de los algoritmos utilizados.

En este cuadro podemos comprobar la escasa eficacia del algoritmo SLUR en la identificación de fórmulas no satisfacibles. Incluso en conjuntos con instancias satisfacibles SOLVELIMILIT($\mathcal{F}$) resuelve un número mayor de fórmulas que el algoritmo anterior. Además de este hecho, se verificó que absolutamente todas las instancias resueltas por el algoritmo SLUR, presentadas en el Cuadro 2.1, son resueltas por SOLVELIMILIT($\mathcal{F}$).

Las fórmulas proposicionales para las cuales el algoritmo SOLVELIMILIT($\mathcal{F}$) finalizó con el mensaje *desisto* fueron testadas con el algoritmo SOLVELIMILIT($\mathcal{F}, 2$) cuyos resultados se muestran en el Cuadro 2.2. Podemos constatar la existencia de dos conjuntos para los cuales no pudo resolverse ninguna instancia. Por otro lado, todas las instancias de los conjuntos *aim*, *cbs*, *jnh*, *rti*, *ssa*, *uf50*, *uuf50*, *uf75*, *uuf75*, *uf100*, *uuf100*, *uf125* y *uuf125* fueron resueltas por este algoritmo.

Además de los tests realizados que presentamos anteriormente, se consideró un conjunto de 1000 rompecabezas del problema SUDOKU. Cada uno de estos rompecabezas tiene solución única que puede ser obtenida usando solamente razonamiento, no prueba y error. Siguiendo la propuesta de Lynce y Ouaknine [LO06], construimos para cada rompecabezas dos fórmulas proposicionales, una según la codificación

Cuadro 2.1: Comparativa de SLUR y SOLVELIMLIT.

Conjunto			SLUR($\mathcal{F}$)			SOLVELIMLIT($\mathcal{F}$)		
Nome	#inst.	r	#Sat	#UnSat	#Desisto	#Sat	#UnSat	#Desisto
aim	72	1.4 a 6.0	2	0	70	48	20	4
bms[1]	500	2.7 a 3.0	4	0	496	39	0	461
cbs[1]	1000	4.0	9	0	991	506	0	494
dubois[2]	12	2.7	0	0	12	0	0	12
jnh	50	8.0 y 9.0	0	0	50	16	33	1
rti[1]	500	4.3	5	0	495	140	0	360
ssa	8	2.2 a 3.3	3	1	4	4	2	2
uf20[1]	1000	4.6	329	0	671	1000	0	0
uf50[1]	1000	4.4	67	0	933	973	0	27
uuf50[2]	1000	4.4	0	0	1000	0	974	26
uf75[1]	100	4.3	4	0	96	55	0	45
uuf75[2]	100	4.3	0	0	100	0	5	95
uf100[1]	1000	4.3	6	0	994	284	0	716
uuf100[2]	1000	4.3	0	0	1000	0	0	1000

([1]) Todas las instancias satisfacibles. ([2]) Todas las instancias no satisfacibles.

Cuadro 2.2: Comparativa SOLVELIMLIT($\mathcal{F}, k$) con $k = 1$ y $k = 2$.

Conjunto		SOLVELIMLIT($\mathcal{F}$)			SOLVELIMLIT($\mathcal{F}, 2$)		
Nome	#inst.	#Sat	#UnSat	#Desisto	#Sat	#UnSat	#Desisto
aim	72	48	20	4	48	24	0
bms[1]	500	39	0	461	275	0	225
cbs[1]	1000	506	0	494	1000	0	0
dubois[2]	12	0	0	12	0	0	12
jnh	50	16	33	1	16	34	0
rti[1]	500	140	0	360	500	0	0
ssa	8	4	2	2	4	4	0
uf50[1]	1000	973	0	27	1000	0	0
uuf50[2]	1000	0	974	26	0	1000	0
uf75[1]	100	55	0	45	100	0	0
uuf75[2]	100	0	5	95	0	100	0
uf100[1]	1000	284	0	716	1000	0	0
uuf100[2]	1000	0	0	1000	0	1000	0
uf125[1]	100	13	0	87	100	0	0
uuf125[2]	100	0	0	100	0	100	0
uf150[2]	100	7	0	93	59	0	41
uuf150[2]	100	0	0	100	0	81	17

([1]) Todas las instancias satisfacibles. ([2]) Todas las instancias no satisfacibles.

mínima, con 8829 cláusulas y otra según la codificación ampliada, con 11988 cláusulas.

Todas las instancias SAT con la codificación ampliada procesadas por el algoritmo ELIMINALITERALES tuvieron como repuesta fórmulas pro-

posicionales constituidas por cláusulas con un único literal. Para éstas, identificar un modelo es una tarea trivial.

Esto no sucede con las fórmulas proposicionales obtenidas con la codificación mínima. De las 1000 fórmulas proposicionales consideradas, tuvieron por repuesta una fórmula con todas sus cláusulas unitarias mediante el algoritmo ELIMINALITERALES($\mathcal{F}$) 116 y mediante el algoritmo ELIMINALITERALES($\mathcal{F}, 2$), 944. Faltaron por resolver 56 del total de instancias consideradas, las cuales aún no se han probado utilizando el algoritmo ELIMINALITERALES($\mathcal{F}, 3$). Todas las fórmulas han sido examinadas con el algoritmo PropUnit, mediante el que no se ha logrado resolver ninguna.

Cuadro 2.3: Rompecabezas SUDOKU. Resolución de instancias SAT con las codificaciones mínima y ampliada.

Codificación	#inst.	ELIMINA-LITERALES($\mathcal{F}$)	ELIMINA-LITERALES($\mathcal{F}, 2$)	PropUnit
Ampliada	1000	1000	—	0
Mínima	1000	116	944	0

2.6. _PURL+, una clase polinomial no sensible a la entrada

Hemos observado anteriormente que la existencia de una cláusula con dos literales p-eliminables puede conducir a distintas fórmulas lógicamente equivalentes (por ejemplo, a partir de la fórmula proposicional (2.3) se obtienen dos distintas fórmulas (2.4a) y (2.4b)). Se comprueba fácilmente que, en general, si $\mathcal{F}^1$ y $\mathcal{F}^2$ son distintas fórmulas lógicamente equivalentes a $\mathcal{F}$, obtenidas por exclusión de literales eliminables, entonces la unión $\mathcal{F}^1 \cup \mathcal{F}^2$ es una fórmula lógicamente equivalente a $\mathcal{F}$. Además, cada modelo de la unión de estas dos fórmulas es también un modelo de cada una de ellas, $\mathcal{F}^1$ y $\mathcal{F}^2$, y por tanto modelo de $\mathcal{F}$ (y viceversa).

Recordemos que, en la fórmula presentada en el ejemplo (2.3),

$$\mathcal{F} = \{[1, 2, 3], [2, 4], [3, \overline{4}], [1, \overline{5}], [\overline{2}, \overline{5}]\}, \tag{2.10}$$

los literales $1 \in [1, 2, 3]$ y $2 \in [1, 2, 3]$ son p-eliminables. Entonces, duplicando la cláusula $[1, 2, 3]$ en $\mathcal{F}$ y, en cada una de ellas, excluyendo uno

de los dos literales p-eliminables, se obtiene la anterior fórmula proposicional correspondiente a la unión $\mathcal{F}^1 \cup \mathcal{F}^2$. Esta perspectiva conduce a otro abordaje sobre las formas de excluir literales p-eliminables, separando el proceso de reconocimiento de los literales p-eliminables de la su exclusión. En una primera fase se investiga en todas las cláusulas la existencia de este tipo de literales (p-eliminables) y se construye una lista de cláusulas y de sus respectivos literales p-eliminables.

En una segunda etapa, cada cláusula conteniendo literales p-eliminables es sustituida por nuevas cláusulas, una por cada literal p-eliminable. Suponiendo, por ejemplo, que $l_1, \cdots, l_t \in C$ son los literales p-eliminables identificados en esta etapa, la cláusula $C \in \mathcal{F}$ es excluida y se añaden t distintas cláusulas, obtenidas de C excluyendo un literal por cada, $\mathcal{F} = (\mathcal{F} - \{C\}) \cup \{C - [l_1]\} \cup \cdots \cup \{C - [l_t]\}\}$.

Algoritmo 6 IDENTIFICAELIMINABLES($\mathcal{F}$)

Entrada: Una CNF-fórmula $\mathcal{F}$ (sin cláusula nula)
Salida: Lista de cláusulas y de sus respectivos literales p-eliminables, $\mathcal{L}$

```
    L = ∅
    for all C ∈ F do
      E = ∅
      for all l ∈ C do
5:      (F', τ') = PROPUNIT(F ∪ U(¬Flip(C, l)))
        if □ ∈ F' then
          E = E ∪ {l}
        end if
      end for
10:   if E ≠ ∅ then
        P = {C, E}, L = L ∪ {P}
      end if
    end for
    return (L)
```

Para la primera etapa se ha desarrollado el algoritmo IDENTIFICAELIMINABLES (v. Algoritmo 6). En él, todas las cláusulas y literales son comprobadas y los p-eliminables registrados en una lista. Dado que para cada cláusula y para cada uno de sus literales el algoritmo de propagación unitaria es ejecutado una vez, el algoritmo IDENTIFICAELIMINABLES se ejecuta en tiempo de orden $O(|\mathcal{F}|^2)$.

En una segunda etapa, para excluir los literales p-eliminables en cada una de las cláusulas, usamos el algoritmo EXCLUYEELIMINABLES (v. Algoritmo 7). La entrada está constituida por una fórmula propo-

Algoritmo 7 EXCLUYEELIMINABLES($\mathcal{F}, \mathcal{L}$)

Entrada: Una CNF-fórmula $\mathcal{F}$ y la lista de cláusulas y sus respectivos literales p-eliminables, $\mathcal{L}$

Salida: La CNF-fórmula lógicamente equivalente con los literales p-eliminables excluidos, $\mathcal{F}^E$.

```
for all P = {C, E} ∈ L do
  F = F − {C}
  for all l ∈ E do
5:  F = F ∪ (C − [l])
    if □ ∈ F then
      F = {□}
      return (F)
    end if
10: end for
end for
return F
```

sicional y una lista $\mathcal{L}$ de cláusulas y sus respectivos literales eliminables. Cada elemento de la referida lista ($\mathcal{P} \in \mathcal{L}$) es un par $\mathcal{P} = \{C, E\}$ constituido por una cláusula, C y por la lista de literales de C p-eliminables. Para cada $\mathcal{P} \in \mathcal{L}$, la fórmula proposicional $\mathcal{F}$ es actualizada, $\mathcal{F} = (\mathcal{F} - C) \cup \{C - l_1\} \cup \cdots \cup \{C - l_t\}$, dónde $l_1, \cdots, l_t \in E$ son literales de C p-eliminables.

En estas condiciones, la fórmula proposicional lógicamente equivalente obtenida en cada iteración podrá crecer en número de cláusulas siempre que en $\mathcal{P}$ existan cláusulas con más de un literal p-eliminable.

Pero durante este proceso se pueden añadir cláusulas repetidas. Si se pretende evitar la existencia de cláusulas duplicadas durante el procesamiento del algoritmo EXCLUYEELIMINABLES, es imprescindible verificar la existencia previa de una nueva cláusula antes de añadirla. Esto podría ser efectuado en tiempo orden $O(log(m))$ (dónde m designa el número de cláusulas) si la lista de cláusulas estuviera ordenada, por ejemplo en orden lexicográfico. Dado que la lista $\mathcal{L}$ contiene a lo sumo $|\mathcal{F}|$ cláusulas y literales, el algoritmo EXCLUYEELIMINABLES corre en tiempo de orden $O(|\mathcal{F}|log(m))$, si no existen cláusulas repetidas.

Con los dos algoritmos anteriores podemos ahora construir un nuevo algoritmo para detectar y excluir literales p-eliminables y p-eliminables ocultos, que designaremos algoritmo ELIMINALITERALES+ (v. Algoritmo 8).

Este algoritmo presenta una estructura muy simple. En cada iteración se invocan los algoritmos IDENTIFICAELIMINABLES y EXCLUYEE-

Algoritmo 8 ELIMINALITERALES+($\mathcal{F}$)

Entrada: Una CNF-fórmula $\mathcal{F}$ (sin cláusula nula)
Salida: La CNF-fórmula sin literales p-eliminables $\mathcal{F}^E$ (lógicamente equivalente a $\mathcal{F}$)

```
repeat
  L = IDENTIFICAELIMINABLES(F)
  if L ≠ ∅ then
    F = EXCLUYEELIMINABLES(F, L)
  end if
until (F = {□}) or (L = ∅)
return (F)
```

LIMINABLES, hasta que la fórmula lógicamente equivalente obtenida contenga la cláusula nula o hasta que no contenga cláusulas con literales p-eliminables.

Al contrario que el algoritmo ELIMINALITERALES($\mathcal{F}$), este nuevo algoritmo, ELIMINALITERALES+($\mathcal{F}$), no depende de la ordenación de las cláusulas y literales de la entrada. La demostración de esta afirmación es una simple comprobación. Sean $\mathcal{F}^{'}$ y $\mathcal{F}^{''}$ dos fórmulas proposicionales con las mismas cláusulas y literales que $\mathcal{F}$, pero con distintas ordenaciones. Para una misma cláusula $C^{'}$ y $C^{''}$ en $\mathcal{F}^{'}$ y $\mathcal{F}^{''}$, respectivamente, cada literal l de $C^{'}$ es también literal en $C^{''}$. Pero, si l es p-eliminable en $\mathcal{F}^{'}$, PROPUNIT($\mathcal{F}^{'} \cup \mathcal{U}(Flip(C^{'}, l))$ retorna una fórmula con la cláusula nula. Lo mismo sucede con PROPUNIT($\mathcal{F}^{''} \cup \mathcal{U}(Flip(C^{''}, l))$ pues $\mathcal{F}^{'}$ tiene las mismas cláusulas y literales que $\mathcal{F}^{''}$. Por lo tanto, en cada iteración del algoritmo ELIMINALITERALES+ con las fórmulas $\mathcal{F}^{'}$ y $\mathcal{F}^{''}$ se obtiene la misma lista de literales p-eliminables. Eliminados los literales eliminables de esta lista en $\mathcal{F}^{'}$ y $\mathcal{F}^{''}$, se obtienen fórmulas idénticas difiriendo a lo sumo en la ordenación de las cláusulas y los literales.

Como se observó anteriormente, en cada iteración del algoritmo ELIMINALITERALES+, si existe una cláusula con más de un literal p-eliminable, el número de cláusulas aumenta.

Recordemos que cada nueva cláusula tiene un literal menos que la cláusula original (substituida). Para determinar una cota superior para el crecimiento de cada una de las fórmulas proposicionales consideremos $C_i^{'}$ una de las cláusulas obtenidas a partir de C_i después de eliminados t literales $S_{ij} = \{l_{j_1}, \cdots, l_{j_t}\}$, no necesariamente por este orden. Como el orden de exclusión no altera a la cláusula obtenida, consideremos que S_{ij} está ordenada por orden creciente en el índice de los literales. El número total de distintas cláusulas derivadas de la cláusula C_i en la fórmula $\mathcal{F}^E$ (lógicamente equivalente a $\mathcal{F}$, obtenida al final del procesamiento) corresponde como mucho al número de secuencias distintas S_{ij} de literales p-eliminables.

Para probar la afirmación anterior, observemos que si $C_i^{'}$ y $C_i^{''}$ son dos cláusulas distintas obtenidas a partir de la cláusula C_i, cuyas secuencias de literales excluidos son respectivamente $S_{ij_1} = \{l_{i_1}, \cdots, l_{i_r}\}$ y $S_{ij_2} = \{l_{j_1}, \cdots, l_{j_s}\}$, entonces ni $S_{ij_1} \subset S_{ij_2}$ ni $S_{ij_2} \subset S_{ij_1}$. Caso contrario, podemos suponer sin pérdida de generalidad que $S_{ij_1} \subset S_{ij_2}$ y tendríamos que $C_i^{''} \subset C_i^{'}$, con lo que los literales de $C_i^{''} - C_i^{'}$ sería p-eliminables y esto entra en contradicción con la hipótesis pues al final del procesamiento del algoritmo la fórmula obtenida $\mathcal{F}^E$ no contiene cláusulas con literales p-eliminables.

En el caso de que todas las secuencia S_{ij} tuviesen el mismo número de literales, digamos t, a lo sumo existiría un número de secuencias igual al número de conjuntos de m_i literales con t elementos, $\binom{m_i}{t}$. Esto coincide con el número de distintas subcláusulas de C_i con $m_i - t$ literales, siendo m_i el número de literales de la cláusula C_i. En estas condiciones, el máximo sucede cuando $t = \lfloor \frac{m_i}{2} \rfloor$. Por el contrario, si existe una secuencia con longitud distinta de t, entonces una de estas secuencias de longitud t no puede existir, por lo que o máximo número de cláusulas que la cláusula C_i puede originar nunca excede de $\binom{m_i}{\lfloor \frac{m_i}{2} \rfloor}$.

Sea ahora $\mathcal{F}^{'}$ la fórmula con mayor número de cláusulas y literales obtenida durante el procesamiento del algoritmo, se tiene que cada cláusula C_i de la fórmula inicial ($\mathcal{F}$) podrá dar origen a un máximo de $\binom{m_i}{\lfloor \frac{m_i}{2} \rfloor}$ nuevas cláusulas con $\lceil \frac{m_i}{2} \rceil$ literales. Considerando que el tamaño de una fórmula corresponde a la suma del número de literales en cada una de sus cláusulas, y que la cláusula inicial $C_i \in \mathcal{F}$ ($i = 1, \cdots, m$) podrá dar origen hasta $\binom{m_i}{\lceil \frac{m_i}{2} \rceil}$ cláusulas, cada una con $\lfloor \frac{m_i}{2} \rfloor$ literales como máximo, tenemos que:

$$|\mathcal{F}^{'}| \leq \sum_{i=1}^{m} \left\lceil \frac{m_i}{2} \right\rceil \binom{m_i}{\lfloor \frac{m_i}{2} \rfloor} \leq \sum_{i=1}^{m} m_i \binom{m_i}{\lfloor \frac{m_i}{2} \rfloor}.$$

Considerando que c es el mayor número de cláusulas al que una cláusula puede dar origen:

$$|\mathcal{F}^{'}| \leq \sum_{i=1}^{m} m_i c = c \sum_{i=1}^{m} m_i = c|\mathcal{F}| = c|\mathcal{F}|.$$

Dado que el algoritmo IDENTIFICAELIMINABLES($\mathcal{F}^{'}$) corre en tiempo cuadrático respecto al tamaño de la entrada $|\mathcal{F}^{'}|$, en el peor de los casos el algoritmo ELIMINALITERALES+($\mathcal{F}^{'}$) se ejecuta en tiempo $O(c^3|\mathcal{F}|^3)$.

Si $\mathcal{F}$ es una instancia 3SAT o 4SAT, $c < 2^3$ y $c < 3^3$, respectivamente.

El algoritmo ELIMINALITERALES+ presenta características que permiten fácilmente una implementación utilizando computación paralela,

pues los literales de cada una de las cláusulas pueden ser comprobados con la fórmula proposicional independientemente de las otras cláusulas.

A partir del algoritmo ELIMINALITERALES+, presentado en la presente sección, y a semejanza de la definición presentada para la clase PURL (v. Definición 2.3), se puede definir una nueva clase PURL+.

Teniendo presente la fórmula devuelta por el algoritmo ELIMINALITERALES($\mathcal{F}$), queda por establecer la relación entre ambas clases: PURL y PURL+.

Se conjetura que, si $\square \in$ ELIMINALITERALES($\mathcal{F}$) entonces $\square \in$ ELIMINALITERALES+($\mathcal{F}$). De comprobarse esta conjetura tendríamos que PURL es una subclase de PURL+.

2.7. Algoritmo casicuadrático para detectar y eliminar literales p-eliminables

Como se puede comprobar fácilmente, el algoritmo ELIMINALITERALES($\mathcal{F}$) (v. Algoritmo 3) no es tan eficiente como pudiera serlo, pues existe una innecesaria repetición de la propagación unitaria de las distintas atribuciones de verdad $\overline{Flip(C,l)}$ en $\mathcal{F}$. Esta situación puede ser evitada guardando las fórmulas proposicionales obtenidas en una iteración para uso posterior, suspendiendo y retomando la propagación unitaria siempre que una cláusula es simplificada. Esto ocurre siempre que un literal p-eliminable es eliminado.

Consideremos, a este efecto, la siguiente versión del algoritmo ELIMINALITERALES($\mathcal{F}$) que, para distinguir, designaremos por ELIMINALITERALES$_1$($\mathcal{F}$) (v. Algoritmo 9).

Durante la ejecución del algoritmo ELIMINALITERALES$_1$($\mathcal{F}$), la línea 4 se ejecuta una vez para cada cláusula C y para cada literal $l \in C$. La fórmula retornada por el algoritmo PROPUNIT($\mathcal{F} \cup \mathcal{U}(\overline{Flip(C,l)})$), $\mathcal{R}(C,l)$, se guarda para una posible utilización posterior. Si esta fórmula contiene la cláusula nula, el literal $l \in C$ es p-eliminable. En este caso el par (C,l) es añadido a una lista $\mathcal{L}$, que será procesada por el algoritmo PROCESALISTA$_1$($\mathcal{L},\mathcal{F}$). Caso contrario, se suspende la propagación unitaria que se retoma cuando alguna de las cláusulas de $\mathcal{F}$ sea simplificada, como veremos a continuación. En el procesamiento de este algoritmo, sin considerar las invocaciones al algoritmo de procesamiento de la lista de literales p-eliminables, se crean a lo sumo $|\mathcal{F}|$ fórmulas $\mathcal{R}(C,l)$, cada una de ellas en tiempo $O(|\mathcal{F}|)$.

Algoritmo 9 ELIMINALITERALES$_1(\mathcal{F})$

Entrada: Fórmula proposicional $\mathcal{F}$ (sin cláusula nula)
Salida: Fórmula sin literales p-eliminables, $\mathcal{F}^E$

```
    L = ∅
    for all C ∈ F do
      for all l ∈ C do
        (R(C,l), τ) = PROPUNIT(F ∪ U(¬Flip(C,l)))
 5:     if □ ∈ R(C,l) then
          L = L ∪ {(C,l)}
          F = PROCESALISTA1(L, F)
        end if
      end for
10: end for
    return (F)
```

En cada invocación del algoritmo PROCESALISTA$_1(\mathcal{L}, \mathcal{F})$ se actualiza la fórmula proposicional $\mathcal{F}$ para cada par (C, l) en $\mathcal{L}$, eliminando el literal p-eliminable $l \in C$ de esta cláusula. Se comprueba que C es una cláusula de $\mathcal{F}$, pues puede haber sido modificada anteriormente debido a otro literal p-eliminable. Por cada literal p-eliminable $l \in C$ se producen $k-1$ nuevas fórmulas $\mathcal{R}(C-[l], u)$, una por cada literal $u \in C-[l]$, (dónde k designa el número de literales de la cláusula C). Para todas las demás fórmulas $\mathcal{R}(D, u)$ no es necesario ejecutar de nuevo el algoritmo de propagación unitaria (línea 15). En efecto, basta simplificar esta cláusula, C, excluyendo el literal l y retomar el algoritmo de propagación unitaria previamente suspenso.

En términos de coste, durante el procesamiento del algoritmo ELIMINALITERALES$_1(\mathcal{F})$, en la línea 4, por cada cláusula C_j pueden ser creadas k_j fórmulas $\mathcal{R}_j^1, \cdots, \mathcal{R}_j^{k_j}$, dónde k_j corresponde al número de literales de esta cláusula. Por lo tanto, se pueden crear a lo sumo $|\mathcal{F}|$ fórmulas que denotaremos $\mathcal{R}_1^1, \cdots, \mathcal{R}_1^{k_1}, \mathcal{R}_2^1, \cdots, \mathcal{R}_2^{k_2}, \cdots, \mathcal{R}_m^1, \cdots, \mathcal{R}_m^{k_m}$.

Consideremos a continuación el algoritmo PROCESALISTA$_1(\mathcal{L}, \mathcal{F})$, al que invocamos siempre que se identifica un literal p-eliminable. Para ello, consideremos el par (C_j, l) en $\mathcal{L}$. En la línea 4 de este algoritmo comenzamos por actualizar la fórmula proposicional $\mathcal{F}$, excluyendo el literal l de la cláusula C_j. Como consecuencia de esta alteración, deben actualizarse cada una de las fórmulas proposicionales $\mathcal{R}_i^t$ en memoria. Para esta actualización hay dos casos distintos a considerar, la actualización de las fórmulas $\mathcal{R}_j^s = \mathcal{R}(C_j, u_s)$, para cada literal $u_s \in C_j$ y la actualización de las fórmulas $\mathcal{R}_i^r = \mathcal{R}(D, u_r)$, con $D \neq C_j$ y $u_r \in D$.

En el primer caso, si $u_s = l$ la fórmula puede ser eliminada de la memoria. Si $u_s \neq l$, la fórmula $R_j^s = R(C_j - [l], u_s)$ se recalcula

Algoritmo 10 PROCESALISTA($\mathcal{L}, \mathcal{F}$)

Entrada: Lista de cláusulas y literales p-eliminables $\mathcal{L}$ y fórmula proposicional $\mathcal{F}$
Salida: Fórmula proposicional $\mathcal{F}$

```
    while L ≠ ∅ do
      (C, l) = PrimeiroElemento(L)
      if C ∈ F then
        F = (F − {C}) ∪ {C − [l]}
 5:     if □ ∈ F then
          F = {□}
          return (F)
        end if
        for all R(D, u) do
10:       if D = C then
            if u ≠ l then
              R(D, u) = R(C − [l], u) = PROPUNIT(F ∪
              U(Flip(C − [l], u)))
            end if
          else
15:         R(D, u) = PROPUNIT(F ∪U(Flip(D, u))
          end if
          if □ ∈ R(D, u) then
            L = L ∪ {(D, u)}
          end if
20:     end for
      end if
    end while
    return (F)
```

en tiempo $O(|\mathcal{F}|)$ por propagación unitaria de la atribución de verdad $\overline{Flip(C_j - [l], u_s)}$ en $\mathcal{F}$ (después de actualizada en la línea 4).

Para el análisis del segundo caso consideremos la fórmula $\mathcal{R}(D, u_r) = \{C_1, \cdots, C_j, \cdots C_m\}$ y la atribución de verdad $\tau = \overline{Flip(D, u_r)} \cup \{l_1, \cdots, l_k\}$ obtenida por propagación unitaria. En estas condiciones, o bien (a) $C_j - [l]$ es una cláusula verdadera para la atribución τ, o bien (b) $C_j - [l]$ no contiene a ninguno de los literales de la atribución τ. En el caso (a) es suficiente actualizar la cláusula C_j en $\mathcal{R}_i^k$, lo que puede ser realizado en tiempo constante, utilizando una lista adecuada de punteros para las cláusulas de $\mathcal{F}$ y los literales respectivos en cada una de estas fórmulas. En el segundo caso, (b), la misma cláusula se actualiza igualmente en $\mathcal{R}_i^k$ y se retoma la propagación unitaria.

Para implementar este algoritmo, es particularmente adecuada la

propuesta de algoritmo de propagación unitaria presentada en 1996 por Zhang y Stickel [ZS96].

Puede concluirse, por tanto, que durante el procesamiento del algoritmo PROCESALISTA$_1$, para cada cláusula C_j del par (C_j, l) en $\mathcal{L}$, serán calculadas $k_j - 1$ nuevas fórmulas $\mathcal{R}(C_j - [l], u)$ en tiempo lineal, $O(|\mathcal{F}|)$ y actualizadas las restantes $\mathcal{R}_s^i$, con $s = 1, \cdots, m$ ($s \neq j$) y $i = 1, \cdots, k_s$, en tiempo constante.

Como cada cláusula C_j puede ser reducida, a lo sumo, hasta una cláusula unitaria, en conjunto el número de nuevas cláusulas no podrá exceder de $\dfrac{k_j(k_j-1)}{2}$.

Por lo tanto, el tiempo total gastado por el algoritmo ELIMINALITERALES$_1(\mathcal{F})$, incluyendo los costes de procesamiento del algoritmo PROCESALISTA$_1$, para una fórmula proposicional $\mathcal{F}$, es de $O((mk)^2 + (mk)^2(\dfrac{k-1}{2}) + (m-1)mk^2)$, dónde k designa el máximo número de literales de las cláusulas de $\mathcal{F}$. Lo que equivale a un tiempo total de $O(k|\mathcal{F}|^2)$.

Para futuras referencias observemos que cada par (C, l), con $l \in C$, podría ser añadido a lo sumo una vez a la lista $\mathcal{L}$. Por lo tanto a esta lista se le puede agregar un máximo de $m(k+1)\dfrac{k}{2}$ elementos.

Como consecuencia de algunas de las observaciones anteriores y considerando las características de los algoritmos ELIMINALITERALES$_1(\mathcal{F})$ y PROCESALISTA$_1$, podemos enunciar el siguiente resultado:

Lema 2.14. *Dada una fórmula proposicional $\mathcal{F}$, el algoritmo* ELIMINALITERALES$_1(\mathcal{F})$ *se ejecuta en tiempo* $O(k|\mathcal{F}|^2)$.

Cuando k es una constante (independiente de la entrada), como por ejemplo en los casos de instancias 3SAT o 4SAT, el algoritmo ELIMINALITERALES$_1(\mathcal{F})$ se ejecuta en tiempo cuadrático. Aunque este algoritmo implementa una metodología de detección y exclusión de literales p-eliminables, de modo similar a como lo hace el algoritmo ELIMINALITERALES($\mathcal{F}$), las fórmulas proposicionales retornadas por los dos algoritmos no tienen porqué coincidir. Esto se debe a que el orden de exclusión de literales no coincide, en general. Es posible, empero, hacer una gestión del proceso de los literales p-eliminables mediante una lista auxiliar $\mathcal{M}$, complementaria a $\mathcal{L}$ y conseguir así que el orden de exclusión sea el mismo en ambos algoritmos.

A este efecto consideremos las cláusulas ordenadas por el índice de posición en la fórmula proposicional, i.e., $C_i < C_j$ si $i < j$ dónde $\mathcal{F} = \{C_1, C_2, \cdots, C_m\}$. Y, para cada cláusula, $C_i = [l_1, \cdots, l_{k_i}]$ el literal $l_r < l_s$ si en la lista de los literales de C_i el literal l_r está a la izquierda del literal l_s. Con estas dos ordenaciones, para las cláusulas de la fórmula proposicional y para los literales en cada una de las cláusulas,

consideremos los dos siguientes algoritmos ELIMINALITERALES$_2(\mathcal{F})$ y PROCESALISTA$_2(\mathcal{L}, \mathcal{M}, \mathcal{F})$.

Algoritmo 11 ELIMINALITERALES$_2(\mathcal{F})$

Entrada: Fórmula proposicional $\mathcal{F}$ (sin cláusula nula)
Salida: Fórmula sin literales p-eliminables, $\mathcal{F}^E$

```
    L = ∅, M = ∅
    for all C ∈ F do
      for all l ∈ C do
        (R(C,l), τ) = PROPUNIT(F ∪ U(¬Flip(C,l)))
 5:     if □ ∈ R(C,l) then
          L = L ∪ {(C,l)}
          F = PROCESALISTA(L, M, F)
        end if
      end for
10: end for
    while (M ≠ ∅) y (□ ∉ F) do
      L = M, M = ∅
      F = PROCESALISTA(L, M, F)
    end while
15: return (F)
```

Comparando el algoritmo ELIMINALITERALES$_2(\mathcal{F})$ con el anterior, ELIMINALITERALES$_1(\mathcal{F})$, se observa fácilmente que la principal diferencia es la descomposición de la lista de literales p-eliminables $\mathcal{L}$ en dos listas $\mathcal{L}$ y $\mathcal{M}$. Si estas listas están ordenadas según las cláusulas de los pares (C_i, l), entonces el orden por el cual los literales p-eliminables son excluidos durante el procesamiento del algoritmo ELIMINALITERALES$_2(\mathcal{F})$ coincide con el orden de eliminación de los mismos literales por el algoritmo ELIMINALITERALES$(\mathcal{F})$ (v. Algoritmo 3). Como consecuencia, las fórmulas retornadas por ambos algoritmos coinciden.

Consideremos el algoritmo PROCESALISTA$_2(\mathcal{L}, \mathcal{M}, \mathcal{F})$, invocado por ELIMINALITERALES$_2(\mathcal{F})$. Sea $(C, l) \in \mathcal{L}$ el primer elemento de la lista. Nótese que, en estas condiciones, al acceder al primer elemento da lista, éste se retira de $\mathcal{L}$. En la línea 4, el literal $l \in C$ es eliminado de su cláusula respectiva, por lo que hay que actualizar todas las fórmulas R_i^k (en memoria). Durante esta actualización puede ocurrir que en alguna $\mathcal{R}_i^k = \mathcal{R}(D, u)$, se produzca la cláusula nula. Entonces $u \in D$ es un literal p-eliminable en $\mathcal{F}$. En este caso puede darse una de las dos siguientes situaciones: (a) $D < C$ o $D = C$ y $u < l$ o (b) el caso contrario, i.e. $D > C$ o $D = C$ e $u > l$, que estudiaremos por separado.

En el caso (a), la eliminación del literal u de la respectiva cláusula

Algoritmo 12 PROCESALISTA$_2(\mathcal{L}, \mathcal{M}, \mathcal{F})$

Entrada: Lista de cláusulas y literales p-eliminables $\mathcal{L}$ y fórmula proposicional $\mathcal{F}$
Salida: Fórmula proposicional $\mathcal{F}$

```
    while L ≠ ∅ do
      (C, l) = PrimeiroElemento(L)
      if C ∈ F then
        F = (F − {C}) ∪ {C − [l]}
 5:     if □ ∈ F then
          F = {□}
          return (F)
        end if
        for all R(D, u) do
10:       if D = C then
            if u ≠ l then
              R(D, u) = R(C − [l], u) = PROPUNIT(F ∪ U(Flip(C − [l], u)))
            end if
          else
15:         R(D, u) = PROPUNIT(F ∪ U(Flip(D, u)))
          end if
          if □ ∈ R(D, u) then
            if D > C o (D = C y u > l) then
              L = L ∪ {(D, u)}
20:         else
              M = M ∪ {(D, u)}
            end if
          end if
        end for
25:   end if
    end while
    return (F)
```

no debe ocurrir inmediatamente. Entonces, se añade el par (D, u) a la lista $\mathcal{M}$. En el caso del algoritmo ELIMINALITERALES$(\mathcal{F})$, tras la exclusión del literal $l \in C$, la exclusión de $u \in D$ sólo puede ocurrir en la siguiente iteración. Recordemos que, en este algoritmo, la identificación y exclusión de literales se hace de modo iterativo. En cada iteración las cláusulas $C_1, C_2, \cdots, C_m$ son recorridas secuencialmente y, para cada cláusula $C_i = [l_1, \cdots, l_{k_i}]$, los literales son recorridos según su posición en la cláusula (de izquierda a derecha).

En el caso (b), el par (D, u) es añadido a la lista $\mathcal{L}$. Esto sólo sucede cuando el algoritmo PROCESALISTA$_2$ es invocado por la línea 13 del

algoritmo ELIMINALITERALES$_2$. Consideremos (C_j, x) el (ahora) primer elemento de la lista $\mathcal{L}$. Como $\mathcal{L}$ es una lista ordenada según las cláusulas, $C < C_j$ o $C = C_j$ y $l < x$, en el primer caso, las cláusulas de $\mathcal{F}$ entre C y C_j no contienen literales p-eliminables y, en el segundo, ninguno de los literales de C entre l e x es p-eliminable. Esto permite establecer un paralelismo con el algoritmo ELIMINALITERALES($\mathcal{F}$). En la misma iteración en la que el literal $l \in C$ es eliminado (en este algoritmo), $x \in C_j$ es el siguiente literal a ser también eliminado.

Considerando que cada literal $l \in C$ sólo puede ser insertado una vez en una de las listas $\mathcal{L}$ o $\mathcal{M}$, y que cada una de estas listas puede tener a lo sumo $m(k+1)\frac{k}{2}$ elementos, crear y mantener la ordenación de esta lista tiene un coste $O\left(m(k+1)\frac{k}{2} log\left(m(k+1)\frac{k}{2}\right)\right)$. Lo que equivale a $O\left(k|\mathcal{F}| log\left(|\mathcal{F}|\right)\right)$. Por lo tanto, en total, el algoritmo ELIMINALITERALES$_2(\mathcal{F})$ corre en tiempo $O(k|\mathcal{F}|^2)$. Considerando este aspecto y el hecho de que dos literales p-eliminables serán excluidos en el mismo orden en los algoritmos ELIMINALITERALES($\mathcal{F}$) y ELIMINALITERALES$_2(\mathcal{F})$, podemos enunciar los siguientes:

Lema 2.15. *Dada una fórmula proposicional $\mathcal{F}$, el algoritmo* ELIMINALITERALES$_2(\mathcal{F})$ *corre en tiempo* $O(k|\mathcal{F}|^2)$.

Lema 2.16. *Considerando una fórmula proposicional $\mathcal{F}$, a fórmula proposicional sin literales p-eliminables $\mathcal{F}^E$ obtenida por el algoritmo* ELIMINALITERALES$_2(\mathcal{F})$ *coincide con la obtenida por el algoritmo* ELIMINALITERALES($\mathcal{F}$).

La metodología seguida en el desarrollo del algoritmo ELIMINALITERALES$_2(\mathcal{F})$ puede ser adoptada para mejorar la eficiencia del algoritmo ELIMINALITERALES$(\mathcal{F}, r)$ (v. Algoritmo 5), utilizado para detectar y excluir literales rp-eliminables. A este efecto consideraremos los algoritmos ELIMINALITERALES$_3(\mathcal{F}, r)$ y PROCESALISTA$_3(\mathcal{L}, \mathcal{M}, \mathcal{F}, r)$.

Para estudiar el coste temporal total del algoritmo ELIMINALITERALES$_3(\mathcal{F}, r)$ procederemos por inducción.

Comencemos por considerar el caso $r = 2$. Durante la ejecución del algoritmo ELIMINALITERALES$_3(\mathcal{F}, 2)$ el algoritmo ELIMINALITERALES$_2(\mathcal{F})$ es invocado en la línea 3, a lo sumo, una vez por cada cláusula $C \in \mathcal{F}$ y literal $l \in C$, por lo que podrían ser creadas hasta $|\mathcal{F}|$ fórmulas $\mathcal{R}(C, l)$. En el caso de que el algoritmo PROCESALISTA$_3(\mathcal{L}, \mathcal{M}, \mathcal{F}, 2)$ no sea ejecutado, como el algoritmo ELIMINALITERALES$_2(\mathcal{F})$ corre en tiempo $O(k|\mathcal{F}|^2)$, este algoritmo, en este caso particular, corre en tiempo $O(k|\mathcal{F}|^3)$.

Algoritmo 13 ELIMINALITERALES$_3(\mathcal{F}, r)$

Entrada: Fórmula proposicional $\mathcal{F}$ y un entero positivo r
Salida: Fórmula sin literales rp-eliminables, $\mathcal{F}^E$

$\mathcal{L} = \emptyset$, $\mathcal{M} = \emptyset$
if $r = 1$ **then**
  $\mathcal{F} =$ ELIMINALITERALES$_2(\mathcal{F})$
  return $(\mathcal{F})$
end if
for all $C \in \mathcal{F}$ **do**
  for all $l \in C$ **do**
    $(\mathcal{R}(C,l), \tau) =$ ELIMINALITERALES$_3(\mathcal{F} \cup \mathcal{U}(\overline{Flip(C,l)}), r-1)$
    if $\square \in \mathcal{R}(C,l)$ **then**
      $\mathcal{L} = \mathcal{L} \cup \{(C,l)\}$
      $\mathcal{F} =$ PROCESALISTA$_3(\mathcal{L}, \mathcal{M}, \mathcal{F}, r)$
    end if
  end for
end for
while ($\mathcal{M} \neq \emptyset$) y ($\square \notin \mathcal{F}$) **do**
  $\mathcal{L} = \mathcal{M}$, $\mathcal{M} = \emptyset$
  $\mathcal{F} =$ PROCESALISTA$_3(\mathcal{L}, \mathcal{M}, \mathcal{F}, r)$
end while
return $(\mathcal{F})$

En cada invocación del algoritmo PROCESALISTA$_3(\mathcal{L}, \mathcal{M}, \mathcal{F}, 2)$, para cada par $(C_j, l) \in \mathcal{L}$, se actualiza la fórmula proposicional $\mathcal{F}$ excluyendo el literal 2p-eliminable $l \in C_j$. A continuación, se actualizan las distintas fórmulas $\mathcal{R}_i^t$ (en memoria). Más específicamente, la fórmula $\mathcal{R}_j^{s_0} = \mathcal{R}(C_j, l)$ es eliminada y, para cada literal $u \in C_j - [l]$, la fórmula $\mathcal{R}_j^s = R(C_j - [l], u)$, para $s = 1, \cdots, k_j = |C - [l]|$, es calculada en tiempo $O(k|\mathcal{F}|^2)$, correspondiente al tiempo de procesamiento del algoritmo ELIMINALITERALES$_2(\mathcal{F} \cup \mathcal{U}(\overline{Flip(C - [l], u)}))$. Para cada una de las demás fórmulas $\mathcal{R}_i^k = \mathcal{R}(D, u)$, con $D \neq C$, la fórmula es calculada nuevamente, $\mathcal{R}(D,u) =$ ELIMINALITERALES$_2(\mathcal{F} \cup \mathcal{U}(\overline{Flip(D,u)}))$.

De este modo, a cada literal $l \in C$ eliminado le corresponde, en el peor caso (cuando cada una de las cláusulas contiene k literales) un coste de $O(kk|\mathcal{F}|^2 + k(m-1)k|\mathcal{F}|^2)$. Dado que por cada cláusula (en la fórmula inicial) se pueden eliminar, como mucho, $k-1$ literales hasta obtener una cláusula unitaria, resulta un coste, de nuevo en el peor caso, de $O(\frac{(k-1)k}{2}(k^2|\mathcal{F}|^2 + (m-1)k^2|\mathcal{F}|^2))$.

La fórmula proposicional $\mathcal{F}$ contiene m cláusulas, por lo que el coste total es de $O(k|\mathcal{F}|^3 + m\frac{(k-1)k}{2}(k^2|\mathcal{F}|^2 + (m-1)k^2|\mathcal{F}|^2))$ (dónde la primera parte corresponde al coste del cálculo inicial de las fórmulas $\mathcal{R}_i^t$). Esto

Algoritmo 14 PROCESALISTA$_3(\mathcal{L}, \mathcal{M}, \mathcal{F}, r)$

Entrada: Listas de cláusulas y literales p-eliminables $\mathcal{L}$ y $\mathcal{M}$, fórmula proposicional $\mathcal{F}$ y entero positivo r
Salida: Fórmula proposicional $\mathcal{F}$

```
    while L ≠ ∅ do
      (C, l) = PrimeiroElemento(L)
      if C ∈ F then
        F = (F − {C}) ∪ {C − [l]}
 5:     if □ ∈ F then
          F = {□}
          return (F)
        end if
        for all R(D, u) do
10:       if D = C then
            if u ≠ l then
              R(D, u) = R(C − [l], u) = ELIMINALITERALES3(F ∪
              U(Flip(C − [l], u)), r − 1)
            end if
          else
15:         R(D, u) = ELIMINALITERALES3(F ∪U(Flip(D, u)), r − 1)
          end if
          if □ ∈ R(D, u) then
            if D > C o (D = C y u > l) then
              L = L ∪ {(D, u)}
20:         else
              M = M ∪ {(D, u)}
            end if
          end if
        end for
25:   end if
    end while
    return (F)
```

equivale a $O(k|\mathcal{F}|^3 + k^3|\mathcal{F}|^3 + k^2|\mathcal{F}|^4)$. En los casos en los que $k \leq m$, el algoritmo ELIMINALITERALES$_3$ se ejecuta en tiempo $O(k^2|\mathcal{F}|^4)$.

A semejanza del estudio presentado anteriormente, la utilización de las listas $\mathcal{L}$ e $\mathcal{M}$ permite controlar el orden de eliminación de los literales 2p-eliminables, en el algoritmo ELIMINALITERALES$_3$, de la misma forma que lo habíamos comprobado en el algoritmo ELIMINALITERALES$_2$.

La complejidad del algoritmo no se ve afectada por el coste asociado a la ordenación de las listas $\mathcal{L}$ o $\mathcal{M}$, pues por cada literal 2p-eliminable $l \in C$, el par (C, l) sólo podrá ser insertado a la vez en una de las listas

$\mathcal{L}$ o $\mathcal{M}$. Como el máximo número de elementos que cualquiera de estas listas puede tener es $k|\mathcal{F}|$, su ordenación y mantenimiento tienen un coste de $O\left(k|\mathcal{F}|log\left(|\mathcal{F}|\right)\right)$.

Admitamos, por hipótesis de inducción, que el algoritmo ELIMINALITERALES$_3(\mathcal{F}, r)$ se ejecuta en tiempo $O(k^r|\mathcal{F}|^{2r})$, para $r \geq 2$. Durante el procesamiento del algoritmo ELIMINALITERALES$_3(\mathcal{F}, r+1)$, el algoritmo ELIMINALITERALES$_3(\mathcal{F}, r)$ puede ser invocado para la creación inicial de las $|\mathcal{F}|$ fórmulas. A través del algoritmo PROCESALISTA$_3$, invocado $m\frac{(k-1)k}{2}k$ veces por la línea 12 y $m\frac{(k-1)k}{2}(m-1)k$ veces por la línea 15. En total, el algoritmo se ejecuta en tiempo $O((|\mathcal{F}| + (k-1)k|\mathcal{F}| + k|\mathcal{F}|^2)(k^r|\mathcal{F}|^{2r}))$, lo que equivale a $O(k^{(r+1)}|\mathcal{F}|^{2(r+1)})$, cuando $k \leq m$.

Antes de concluir, observemos que el tamaño de la fórmula proposicional sin literales rp-eliminables $\mathcal{F}^E$ nunca excede del tamaño de la fórmula inicial $|\mathcal{F}^E| \leq |\mathcal{F}|$. Es decir, la complejidad establecida es una cota superior de la complejidad del algoritmo.

Finalizamos enunciando el siguiente par de resultados:

Teorema 2.17. *Sea $\mathcal{F}$ una fórmula proposicional. El algoritmo* ELIMINALITERALES$_3(\mathcal{F}, r)$ *se ejecuta en tiempo $O(k^r|\mathcal{F}|^{2r})$, para $r \geq 2$.*

Demostración. La demostración, por inducción, resulta de los comentarios que preceden al enunciado del teorema. □

Teorema 2.18. *Considerando una fórmula proposicional $\mathcal{F}$, la fórmula proposicional sin literales rp-eliminables $\mathcal{F}^E$ obtenida por el algoritmo* ELIMINALITERALES$_3(\mathcal{F}, r)$ *coincide con la obtenida por el algoritmo* ELIMINALITERALES$(\mathcal{F}, r)$.

Demostración. El resultado es una consecuencia del hecho de que los mismos literales rp-eliminables de $\mathcal{F}$ se excluyen en el mismo orden en ambos algoritmos ELIMINALITERALES$_3(\mathcal{F}, r)$ y ELIMINALITERALES$(\mathcal{F}, r)$. □

CAPÍTULO 3

Problema de etiquetado de puntos alineados UC-4P

Una de las principales tareas en la elaboración de mapas es el etiquetado de estos. Es decir, la colocación de texto con información relativa a elementos de naturaleza puntual, lineal o de área. Es un proceso complejo siguiendo los métodos clásicos de producción de cartografía y difícil de automatizar, lo que complica su incorporación en los modernos programas de Cartografía Numérica y Sistemas de Información de Información Geográfica. El tema ha recibido por parte del Computational Geometry Task Force Report [cha96, CET99] el reconocimiento como área importante de investigación dentro del dominio de la Geometría Computacional.

El problema general de etiquetado es intratable [FW91]. Por lo tanto, es importante estudiar casos particulares. Uno destacable es el caso en que los puntos de inserción de las etiquetas están alineados, debido a sus múltiples aplicaciones (representaciones lineales, mapas de metro o ferrocarril, etc). La tesis de doctorado de Reyes estudia un ramillete de problemas de etiquetado y, entre ellos, presenta un algoritmo incremental que permite la resolución de problemas de decisión en el etiquetado de puntos alineados con etiquetas rectangulares con sus lados paralelos a los ejes coordenados, en posiciones fijas o deslizantes [rey02].

Para el caso particular en el que la línea de soporte de los puntos está en posición oblicua respecto a los ejes coordenados, la entrada está formada por n etiquetas de m tamaños distintos ($m \leq n$) y las etiquetas están posicionadas de modo que uno de sus vértices coincide con el punto de inserción de la etiqueta, el problema de existencia de solución puede ser resuelto por el algoritmo incremental en tiempo $O(n \cdot 2^{cm})$, dónde c es una constante independiente de m y de n. En el caso particular de que todas las etiquetas sean de diferentes dimensiones el algoritmo tiene

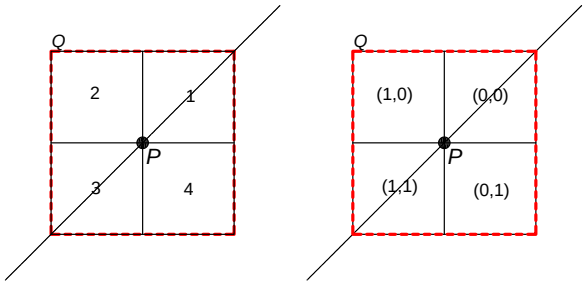


Figura 3.1: Codificación de las cuatro posiciones de una etiqueta en una entrada del problema UC-4P.

complejidad exponencial.

En el presente capítulo estudiaremos el caso particular del problema de etiquetado de puntos alineados mediante una recta con pendiente unidad y cuyas etiquetas sean cuadradas, UC-4P. Para su resolución comenzaremos por reducir sus entradas a instancias del problema de satisfacibilidad. Como principal resultado estableceremos que la clase de satisfacibilidad obtenida, UC-4P-SAT, constituye una subclase de PURL. En particular, veremos que sus instancias pueden ser resueltas en tiempo polinomial por el algoritmo ELIMINALITERALES presentado en el Capítulo 2. En la resolución será utilizado el concepto de grafo de interacciones, el cual introduciremos en su momento. Complementariamente, veremos que UC-4P-SAT no es subclase de ninguna de las clases de satisfacibilidad polinomiales bien conocidas hasta ahora. Por tanto, dado que PURL contienen todas a las clases polinomiales bien conocidas, se constata de nuevo que constituye una extensión de aquellas clases.

3.1. Reducción al problema de satisfacibilidad (Clase UC-4P-SAT)

Dado un punto P en el plano y una etiqueta cuadrada de una entrada del problema UC-4P, ésta puede ocupar cuatro distintas posiciones de modo que uno de sus vértices coincida con P. Numeraremos las respectivas posiciones de 1 a 4 y designaremos cada una de las posiciones por Q^k, $k = 1, \ldots, 4$ y al cuadrado correspondiente al conjunto de las cuatro posiciones por Q, conforme a la representación mostrada en la Figura 3.1 (izquierda).

Formalmente, el problema de decisión UC-4P puede ser enunciado del siguiente modo:

UC-4P

Entrada: Un conjunto de puntos en el plano $\mathcal{P} = \{P_1, \cdots, P_n\}$ sobre una recta (oblícua) de pendiente unidad y un conjunto de etiquetas cuadradas, $\mathcal{L} = \{L_1, \cdots, L_n\}$ con sus respectivos lados paralelos al sistema de ejes coordenados.

Pregunta: ¿Se puede colocar la etiqueta L_i, con alguno de sus vértices en el punto P_i sin que dos etiquetas se intersequen?

En el presente trabajo consideraremos los puntos ordenados de abajo hacia arriba en cada entrada, de modo que $P_i < P_j$ si $i < j$. Caso contrario, los respectivos puntos podrían ser ordenados a partir de una algoritmo de ordenación de una lista de números en tiempo de orden $O(n \cdot log(n))$, dónde $n = |\mathcal{P}|$.

La reducción de ciertos problemas de decisión al problema de la satisfacibilidad es una metodología frecuentemente utilizada. Bien para su resolución, aprovechando el conocimiento actual del problema SAT, bien para su estudio, análisis y establecimiento de propiedades. De un modo general, para establecer este tipo de reducción es indispensable desarrollar un algoritmo eficiente que, en tiempo polinomial, permita convertir cada una de las entradas del problema de decisión en una instancia de SAT.

A continuación presentaremos una metodología que permite reducir cada una de las entradas del problema UC-4P a una instancia del problema de satisfacibilidad en tiempo cuadrático respecto al número de puntos de la entrada.

Consideremos un punto P de una entrada del problema de UC-4P. Las cuatro posiciones de la respectiva etiqueta, Q^k, $k = 1, \ldots, 4$, pueden ser representadas por el par de variables lógicas (l, b), dónde l toma el valor verdadero 1 si la etiqueta ocupa una de las dos posiciones a la izquierda del punto y b, el valor verdadero 1 si la etiqueta ocupa una de las posiciones bajo el punto. La Figura 3.1 (derecha) muestra las cuatro posibles posiciones de una etiqueta y los correspondientes valores de las variables booleanas (l, b) (representando unívocamente cada posición).

Con esta definición para la codificación de las posiciones de las etiquetas, se tiene que si existe una superposición (incluso parcial) de cuadrados, Q_i y Q_j de un par de puntos P_i y P_j aparecen restricciones en el posicionamiento de las etiquetas L_i y L_j, respectivamente. Estas restricciones pueden ser codificadas en una fórmula proposicional que depende de las posiciones relativas de los puntos y de las dimensiones de sus etiquetas. A título de ejemplo veamos el caso en que el punto P_i está en el interior del cuadrado Q_j y el punto P_j en el interior de Q_i (v. Figura 3.2, configuración IV). En esta configuración, las posiciones para las cuales

no hay superposición de las etiquetas corresponden a cada una de las soluciones de la ecuación lógica siguiente:

$$\overline{(\overline{l_i} \wedge \overline{b_i}) \vee (l_j \wedge b_j) \vee (l_i \wedge \overline{b_i} \wedge l_j) \vee (l_i \wedge b_i \wedge l_j \wedge b_j) \vee (\overline{l_i} \wedge b_i \wedge b_j)} = 1$$

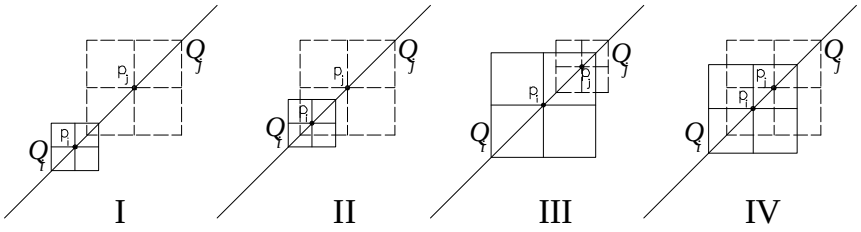


Figura 3.2: Posiciones relativas de un par de puntos P_i y P_j ($P_i < P_j$) y de sus respectivos cuadrados Q_i y Q_j con superposición. Representaciones de las 4 configuraciones topológicamente distintas.

Simplificandola, utilizando las leyes de De Morgan, se obtiene la siguiente fórmula proposicional lógicamente equivalente en forma normal conjuntiva:

$$\mathcal{F}_{i,j} = \{[l_i, b_i], [l_i, \overline{b}_j], [b_i, \overline{l}_j], [\overline{l}_j, \overline{b}_j]\}$$

Las atribuciones de verdad que satisfacen $\mathcal{F}_{i,j}$ corresponden a las posiciones de las etiquetas sin superposición, i.e., a las soluciones del problema.

Analizando todas las posiciones relativas topológicamente distintas de un par de puntos en función de las correspondientes etiquetas, las posibles codificaciones se reducen a cuatro fórmulas proposicionales elementales, además de la fórmula vacía cuando no hay superposición entre los respectivos cuadrados Q_i y Q_j. La Figura 3.2 muestra las referidas posiciones, numeradas de I a IV y el Cuadro 3.1 las correspondientes fórmulas proposicionales en forma normal conjuntiva.

Para cada par de puntos, P_i y P_j de una entrada del problema UC-4P, designaremos por *fórmula proposicional elemental* a la fórmula proposicional referida en el Cuadro 3.1 que corresponde a la posición relativa de ese par de puntos y sus respectivos cuadrados Q_i y Q_j. En el caso de que no haya restricciones al posicionamiento de las etiquetas, la fórmula proposicional elemental es vacía y se omite. De este modo, a cada instancia del problema UC-4P le corresponde una fórmula proposicional en la forma normal conjuntiva que es la conjunción de las fórmulas proposicionales elementales obtenida para cada par de puntos de la entrada original. Por tanto, la reducción de una entrada del problema de

Cuadro 3.1: Fórmulas proposicionales elementales asociadas a las configuraciones representadas en la Figura 3.2 ($i < j$).

	$\mathcal{F}_{i,j}$
I	$\{[l_i, b_i, \bar{l}_j, \bar{b}_j]\}$
II	$\{[\bar{l}_j, \bar{b}_j]\}$
III	$\{[l_i, b_i]\}$
IV	$\{[l_i, b_i], [l_i, \bar{b}_j], [b_i, \bar{l}_j], [\bar{l}_j, \bar{b}_j]\}$

decisión UC-4P al problema de satisfacibilidad puede ser realizada en tiempo cuadrático respecto al número de puntos, analizando las posiciones relativas de los puntos y de sus cuadrados respectivos, tomados dos a dos.

Designaremos por UC-4P-SAT a la familia formada por las fórmulas proposicionales correspondientes a alguna entrada del problema de etiquetado UC-4P. UC-4P-SAT constituye, obviamente, una clase de satisfacibilidad.

§ 3.1.1. Grafo de interacciones.— Cuando se efectúan las reducciones al problema de satisfacibilidad, algunas de las propiedades asociadas a la estructura específica del problema original pueden ser difíciles de identificar, considerando solamente las fórmulas proposicionales obtenidas. El concepto de grafo de interacciones surge como respuesta a esta dificultad. Está especialmente bien adaptado a los problemas de naturaleza geométrica, como es el caso de los problemas de etiquetado y de emparejamiento ortogonal que estudiamos en esta memoria, pero el concepto puede ser fácilmente generalizado.

Dada una entrada del problema UC-4P, $(\mathcal{P}, \mathcal{L})$, designaremos por *grafo de interacciones* de la referida entrada al grafo $G = (V, A)$, construido de modo que a cada punto $P_i \in \mathcal{P}$ le corresponde el vértice $v_i \in V$. La arista $\{v_i, v_j\} \in A$ si y sólo si los cuadrados Q_i y Q_j presentan alguna superposición o,dicho de otra forma, si la fórmula proposicional elemental $\mathcal{F}_{i,j}$ es no vacía. De este modo, se establece una correspondencia entre cada entrada del problema UC-4P, $(\mathcal{P}, \mathcal{L})$, el grafo de interacciones y la fórmula proposicional $\mathcal{F}$ asociando a cada vértice v_i el par de variables booleanas (l_i, b_i) y a cada una de las aristas $\{v_i, v_j\}$ la fórmula elemental proposicional $\mathcal{F}_{i,j}$ (v. diagrama en la Figura 3.3).

El conjunto de vértices del grafo de interacciones se considera ordenado mediante la ordenación inducida por el conjunto de puntos de la entrada, i.e. $v_i < v_j$ si y sólo si $P_i < P_j$.

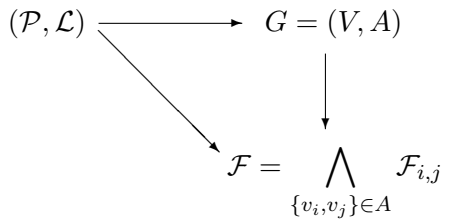


Figura 3.3: Una entrada del problema UC-4P, $(\mathcal{P}, \mathcal{L})$, el grafo de interacciones y la fórmula proposicional correspondiente.

§ 3.1.2. UC-4P-SAT no es subclase de las clases de satisfacibilidad polinomiales bien conocidas.— Vimos anteriormente que las clases de satisfacibilidad polinomiales SLUR y LinAut son incomparables entre si y, en conjunto, contienen todas a las demás clases de satisfacibilidad bien conocidas (v. Sección 1.7 y, específicamente, la Figura 1.17). Por lo tanto, para comprobar que la clase de satisfacibilidad UC-4P-SAT no es subclase de ninguna de las clases polinomiales bien conocidas es suficiente comprobar que existe una instancia de esta clase que no pertenece ni a SLUR ni a LinAut.

A este efecto, consideremos la entrada del problema de etiquetado $(\mathcal{P}, \mathcal{L})$ cuyo grafo de interacciones se indica en la Figura 3.4. Los caracteres romanos junto a cada una de las aristas representan el número de la fórmula proposicional elemental asociada al par de puntos correspondiente. La fórmula proposicional obtenida por reducción de la entrada $(\mathcal{P}, \mathcal{L})$ al problema de satisfacibilidad se indica en (3.1).

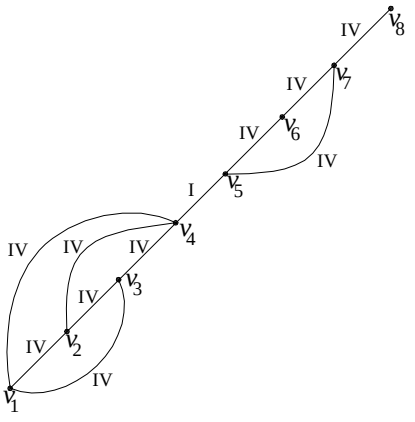


Figura 3.4: Grafo de interacciones de la instancia del problema UC-4P en (3.1).

$$\begin{aligned}\mathcal{F} = \quad & \{[l_1, b_1], [l_1, \overline{b_2}], [b_1, \overline{l_2}], [\overline{l_2}, \overline{b_2}], [l_1, b_1], [l_1, \overline{b_3}], [b_1, \overline{l_3}], [\overline{l_3}, \overline{b_3}], \\ & [l_1, b_1], [l_1, \overline{b_4}], [b_1, \overline{l_4}], [\overline{l_4}, \overline{b_4}], [l_2, b_2], [l_2, \overline{b_3}], [b_2, \overline{l_3}], [\overline{l_3}, \overline{b_3}], \\ & [l_2, b_2], [l_2, \overline{b_4}], [b_2, \overline{l_4}], [\overline{l_4}, \overline{b_4}], [l_3, b_3], [l_3, \overline{b_4}], [b_3, \overline{l_4}], [\overline{l_4}, \overline{b_4}], \\ & [l_4, b_4, \overline{l_5}, \overline{b_5}], \\ & [l_5, b_5], [l_5, \overline{b_6}], [b_5, \overline{l_6}], [\overline{l_6}, \overline{b_6}], [l_5, b_5], [l_5, \overline{b_7}], [b_5, \overline{l_7}], [\overline{l_7}, \overline{b_7}], \\ & [l_6, b_6], [l_6, \overline{b_7}], [b_6, \overline{l_7}], [\overline{l_7}, \overline{b_7}], [l_7, b_7], [l_7, \overline{b_8}], [b_7, \overline{l_8}], [\overline{l_8}, \overline{b_8}]\}\end{aligned} \tag{3.1}$$

Si consideramos ahora el algoritmo SLUR (v. Algoritmo 2), la secuencia de variables l_2, l_5 y escogemos en la opción arbitraria siempre los literales negados, este algoritmo termina con el mensaje *desisto*. Como consecuencia inmediata, $\mathcal{F}$ no es una instancia de la clase SLUR.

$$A = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1
\end{bmatrix} \tag{3.2}$$

Por otro lado, considerando la matriz cláusula-variable de la fórmula proposicional $\mathcal{F}$ (3.1), existe una *autarquía lineal* no nula, $\mathrm{x} \in \mathbb{Q}^n$, satisfaciendo la condición $A\mathrm{x} \geq 0$. Esto acontece considerando valores para $x_1, x_2 > 0$ y $x_{15}, x_{16} < 0$ y los restantes valores todos nulos. La atribución asociada (*asignación de autarquía*) $\tau_1 = \{l_1, b_1, \overline{l_8}, \overline{b_8}\}$, satisface todas las cláusulas de la subfórmula

$$\mathcal{F}_1 = \{[l_1, b_1], [l_1, \overline{b_3}], [b_1, \overline{l_3}], [l_1, b_1], [l_1, \overline{b_4}], [b_1, \overline{l_4}], [l_7, \overline{b_8}], [b_7, \overline{l_8}], [\overline{l_8}, \overline{b_8}]\}$$

y, además, no afecta a ninguna de las cláusulas de $\mathcal{F}_2 = \mathcal{F} - \mathcal{F}_1$. Por lo tanto $\mathcal{F}$ pertenece a LinAut si y sólo si $\mathcal{F}_2$ está en LinAut.

Consideremos ahora el vector $\mathbf{x} = (x_3, \cdots, x_{14}) \in \mathbb{Q}^{12}$ y la matriz cláusula-variable A_2 de la fórmula proposicional $\mathcal{F}_2$. A partir de las inecuaciones $x_3 + x_4 \geq 0$ y $-(x_3 + x_4) \geq 0$, concluimos que $x_3 = -x_4$. Considerando los pares de inecuaciones en idénticas condiciones, podemos concluir que $x_5 = -x_6$, $x_7 = x_8$, $x_{11} = -x_{12}$ y $x_{13} = -x_{14}$. A partir de la inecuación $x_7 + x_8 - x_9 - x_{10} \geq 0$, sabiendo que $-(x_7 + x_8) \geq 0$ y que $x_9 + x_{10} \geq 0$, sumando ordenadamente y conjugando con la anterior, resulta que $x_7 + x_8 = x_9 + x_{10}$ y por tanto $x_9 = -x_{10}$. Por otro lado, conjugando el par de inecuaciones $x_4 - x_5 \geq 0$ y $x_3 - x_6 \geq 0$ resulta la condición $x_4 = -x_6$. Escogiendo los pares adecuados podemos concluir que, $x_4 = -x_8$, $x_6 = -x_8$, $x_{10} = -x_{12}$, $x_{10} = -x_{14}$ y $x_{12} = -x_{14}$. En estas condiciones $x_{12} = 0$ y $x_8 = 0$, así como todos los demás valores $x_i \in \mathbf{x}$ por lo que $\mathbf{x} = 0$. Se concluye, por tanto, que la ecuación $A_2\mathbf{x} \geq 0$ tiene como única solución $\mathbf{x} = \mathbf{0}$.

Dado que la fórmula $\mathcal{F}_2 \subset \mathcal{F}$ (no satisfacible) no pertenece a 2SAT (contiene una cláusula con cuatro literales) y que la condición $A_2\mathbf{x} \geq 0$ no admite ninguna *autarquía lineal*, $\mathbf{x} \in \mathbb{Q}^n$ no nula, se puede concluir que la fórmula proposicional no pertenece a la clase de satisfacibilidad LinAut.

Un comentario final para observar que la fórmula proposicional equivalente devuelta por el algoritmo ELIMINALITERALES($\mathcal{F}$) contiene la cláusula nula, por lo que $\mathcal{F}$ no es satisfacible. Por lo tanto, la instancia $(\mathcal{P}, \mathcal{L})$ tiene respuesta negativa.

3.2. UC-4P-SAT *es subclase de* PURL

En la presente sección demostraremos que UC-4P-SAT es una subclase de la nueva clase de satisfacibilidad polinomial PURL. Veremos también que cualquiera de sus instancias puede ser resuelta en tiempo polinomial por el algoritmo ELIMINALITERALES. A este efecto, comenzaremos por fijar algunas definiciones y establecer los resultados previos necesarios para la demostración del resultado pretendido.

Para ello, dadas una entrada del problema UC-4P con respuesta negativa, la correspondiente fórmula proposicional y el respectivo grafo de interacciones, caracterizaremos la estructura de las fórmulas no satisfacibles mínimas concluyendo la sección con el resultado general sobre la satisfacibilidad de la clase UC-4P-SAT.

§ 3.2.1. Definiciones y resultados previos.— A partir de ahora consideraremos $(\mathcal{P}, \mathcal{L})$, una entrada del problema UC-4P; $\mathcal{F}$, la correspon-

diente fórmula proposicional y $G = (V, A)$, el grafo de interacciones. Para un subconjunto de vértices, $V_1 \subset V$, notaremos $\mathcal{F}_{G_1}$ a la subfórmula asociada al subgrafo de interacciones inducido $G(V_1)$, i.e. $\mathcal{F}_{i,j} \subset \mathcal{F}_{G_1}$ si y sólo si $v_i, v_j \in V_1$ y $\{v_i, v_j\} \in A(G)$. Para simplificar la notación, siempre que no afecte a la claridad del razonamiento, notaremos $\mathcal{F}_{i,j,k}$ en vez de $\mathcal{F}_{G(V_1)}$ en el caso particular $V_1 = \{v_i, v_j, v_k\}$. Del mismo modo, τ_{V_1} designará una atribución de verdad sobre los literales asociados a los vértices del conjunto V_1 y, si $V_1 = \{v_i, v_j, v_k\}$, la referida atribución podrá ser representada por τ_{ijk}.

Dada una fórmula proposicional $\mathcal{F}$ y una atribución de verdad (parcial) τ, consideremos la propagación unitaria $(\mathcal{F}', \tau') = \text{PROPUNIT}(\mathcal{F} \cup \mathcal{U}(\tau))$. En el caso de que $\mathcal{F}'$ no contenga la cláusula nula, utilizaremos la expresión, $\tau \Rrightarrow \tau^1$, para decir que la atribución de verdad $\tau^1 \subset \tau'$ está inducida por propagación unitaria de la atribución τ (en $\mathcal{F}$) o, equivalentemente, que τ^1 es una *consecuencia* de la atribución τ.

Existen instancias particulares de la clase UC-4P-SAT para las cuales es fácil comprobar que son también instancias de PURL. Por ejemplo, si $\mathcal{F}$ no contiene ninguna fórmula elemental I es una instancia 2SAT y, por lo tanto, de PURL (2SAT $\subset$ PURL).

En otro sentido, observemos que si $\mathcal{F}$ no contiene ninguna fórmula elemental IV, cualquiera de las atribuciones de verdad $\tau_1 = \{l_i, \overline{b_i}\}$ y $\tau_2 = \{\overline{l_i}, b_i\}$, $i = \{1, \ldots, n\}$ ($n = |\mathcal{P}|$), constituye un modelo de $\mathcal{F}$. Por lo tanto, en estas condiciones, cada una de las cláusulas $C \in \mathcal{F}$, $C = [l_i, b_i]$, $C = [\overline{l_i}, \overline{b_i}]$ o $C = [l_i, b_i, \overline{l_j}, \overline{b_j}]$, es verdadera para la atribución τ_i ($i = 1, 2$). Se deduce que cualquier fórmula proposicional de la clase UC-4P-SAT no satisfacible contiene una fórmula elemental IV. Además, la existencia de ciertas estructuras involucrando esta fórmula elemental tiene un interés particular para el estudio de esta clase.

Lema 3.1. *Sea G un grafo de interacciones y K_3 un su subgrafo inducido por el subconjunto de vértices $\{v_i, v_r, v_s\}$ tal que a cada una de sus aristas le corresponde una fórmula elemental* IV. *Suponiendo sus vértices ordenados, $v_i < v_r < v_s$, se verifica que, para cualquier modelo τ de la fórmula proposicional asociada $\mathcal{F}_{i,r,s}$, $\{l_i, b_i\} \in \tau$ o $\{\overline{l_s}, \overline{b_s}\} \in \tau$.*

Demostración. Sea τ un modelo de la fórmula proposicional $\mathcal{F}_{i,r,s}$. Por hipótesis, $\mathcal{F}_{i,r} = \{[l_i, b_i], [l_i, \overline{b_r}], [b_i, \overline{l_r}], [\overline{l_r}, \overline{b_r}]\} \subset \mathcal{F}$, pues a la arista $\{v_i, v_r\}$ le corresponde una fórmula elemental IV. Por lo tanto, $\{\overline{l_i}\} \Rrightarrow \{b_i, \overline{b_r}\}$ y $\{\overline{b_i}\} \Rrightarrow \{l_i, \overline{l_r}\}$. Suponiendo que no ocurre la atribución $\{l_i, b_i\}$ ($\subset \tau$), uno de los literales $\overline{l_i}$ o $\overline{b_i}$ es un literal en τ. Sin pérdida de generalidad, podemos suponer que $\overline{l_i} \in \tau$, $\{\overline{l_i}\} \Rrightarrow \{b_i, \overline{b_r}, \overline{b_s}, l_r, \overline{l_s}\}$, pues $\mathcal{F}_{i,s}$ y $\mathcal{F}_{r,s}$ son fórmulas elementales IV. Se deduce que $\{\overline{l_s}, \overline{b_s}\} \in \tau$. □

Como consecuencia del lema anterior, se obtiene el siguiente

Lema 3.2. *Sea K_3 un subgrafo del grafo de interacciones, $G = (V, A)$, inducido por el subconjunto de vértices $\{v_i, v_r, v_s\}$. Si $v_i < v_r < v_s$, $\mathcal{F}_{i,s}$ es una fórmula elemental* IV*, τ constituye un modelo de $\mathcal{F}_{i,r,s}$ y $\{l_i, \overline{b_i}, \overline{l_s}, b_s\} \subset \tau$, entonces $\mathcal{F}_{i,r}$ es una fórmula elemental* III *o $\mathcal{F}_{r,s}$, una fórmula elemental* II*.*

Demostración. Suponiendo que $\{l_i, \overline{b_i}, \overline{l_s}, b_s\} \subset \tau$, las fórmulas elementales asociadas a las aristas del subgrafo K_3 no pueden ser todas IV, de acuerdo con el Lema 3.1. Dado que a la arista $\{v_i, v_s\}$ le corresponde una fórmula IV, el punto P_i está en el interior del cuadrado Q_s. Como por hipótesis $P_i < P_r < P_s$, el punto P_r está también en el interior de Q_s. Por lo tanto, a la arista $\{v_r, v_s\}$ le corresponde una fórmula elemental II o IV. Por simetría, podemos concluir que el punto P_r está también en el interior del cuadrado Q_i, por lo que a la arista $\{v_i, v_r\}$ le corresponde una fórmula elemental III o IV.

Como no todas las fórmulas pueden ser IV, se deduce el resultado enunciado. □

El resultado del Lema 3.1, válido para un ciclo de longitud 3 del grafo de interacciones, admite una generalización a ciclos de longitud impar superior a tres, en las condiciones de la Figura 3.5, como muestra el siguiente resultado:

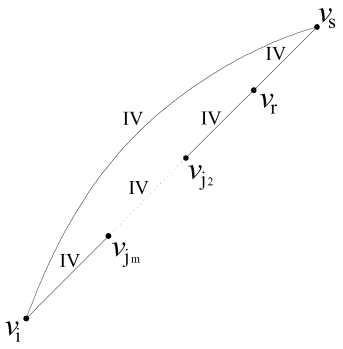


Figura 3.5: Ciclo de longitud impar en un grafo de interacciones con todas las aristas tipo IV (v. Lema 3.3).

Lema 3.3. *Consideremos el ciclo, $C = \{v_i, v_{j_m}, \ldots, v_{j_2}, v_r, v_s, v_i\}$, de longitud impar igual o superior a tres, en el grafo de interacciones, $G = (V, A)$, dónde $v_i < v_{j_m} < \ldots < v_{j_1} < v_r < v_s$ y a cada una de sus aristas le corresponde una fórmula elemental* IV *(v. Figura 3.5). Si τ es un modelo para la fórmula proposicional asociada al referido ciclo, $\mathcal{F}_C$, entonces $\{l_i, b_i\} \subset \tau$ o $\{\overline{l_k}, \overline{b_k}\} \subset \tau$.*

Demostración. La demostración de este resultado sigue el mecanismo de la demostración del Lema 3.1, atendiendo a que, para tres vértices consecutivos, digamos $v_{i_3} < v_{i_2} < v_{i_1}$ en C, $\{\overline{l_{i_3}}\} \Rrightarrow \{\overline{b_{i_2}}, l_{i_2}, \overline{l_{i_1}}\}$ y $\{\overline{b_{i_3}}\} \Rrightarrow \{\overline{l_{i_2}}, b_{i_2}, \overline{b_{i_1}}\}$.

Así, suponiendo que $\{l_i, b_i\} \not\subset \tau$, $\overline{l_i} \in \tau$ o $\overline{b_i} \in \tau$. En el primer caso, $\{\overline{l_i}\} \Rrightarrow \{b_i, l_{j_m}, \overline{b_{j_m}}, \ldots, \overline{l_{j_2}}, b_{j_2}, l_r, \overline{b_r}, \overline{l_s}, \overline{b_s}\}$ y, en el segundo, $\{\overline{b_i}\} \Rrightarrow \{l_i, b_{j_m}, \overline{l_{j_m}}, \ldots, \overline{b_{j_2}}, l_{j_2}, b_r, \overline{l_r}, \overline{l_s}, \overline{b_s}\}$. Por tanto, en cualquier caso $\{\overline{l_s}, \overline{b_s}\} \subset \tau$. □

Para concluir el presente párrafo presentamos un último e importante resultado. En el caso de que el grafo de interacciones de una entrada del problema UC-4P contenga una secuencia de vértices, $v_i < v_j < v_r < v_s$ y dos aristas en las condiciones de la Figura 3.6 (izquierda), entonces contiene el grafo de la derecha de la misma figura como subgrafo, como se muestra en el siguiente resultado:

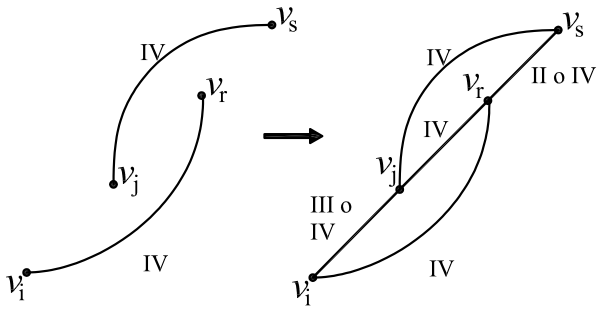


Figura 3.6: Si el grafo de interacciones de una instancia UC-4P contiene las aristas de la izquierda entonces contiene al grafo de la derecha como subgrafo (v. Lema 3.4).

Lema 3.4. *Consideremos el grafo de interacciones, $G = (V, A)$, y $v_i < v_j < v_r < v_s$ un subconjunto de vértices de V tal que $\mathcal{F}_{i,r}$ y $\mathcal{F}_{j,s}$ son fórmulas elementales de tipo* IV. *Entonces, se verifican:*

1) *la arista $\{v_j, v_r\} \in A$ y $\mathcal{F}_{j,r}$ es una fórmula elemental* IV,

2) *$\{v_i, v_j\}$ y $\{v_r, v_s\}$ son aristas de A.*

3) *una de las fórmulas $\mathcal{F}_{i,j}$ o $\mathcal{F}_{r,s}$ es una fórmula elemental* IV.

4) $\mathcal{F}_{i,j}$ *es una fórmula elemental* III *o* IV.

5) $\mathcal{F}_{r,s}$*,* II *o* IV.

Demostración. Dado que, por hipótesis, $\mathcal{F}_{i,r}$ es una fórmula elemental IV y $P_i < P_j < P_r$, resulta que el punto P_j está en el interior de los cuadrados Q_i y Q_r, por lo que $\{v_i, v_j\}$ y $\{v_j, v_r\}$ son aristas del grafo de interacciones G. Por simetría, cómo $\mathcal{F}_{j,s}$ es fórmula elemental del tipo IV, P_r está simultáneamente en el interior de los cuadrados Q_j y Q_s, por lo que $\{v_r, v_s\} \in A$. Falta, por tanto, identificar las fórmulas elementales que les corresponden. Como vimos, el punto P_j está en el interior del cuadrado Q_r y P_r en el interior de Q_j. Por tanto, $\mathcal{F}_{j,r}$ es una fórmula elemental IV, lo que permite afirmar las tesis 1) y 2) del lema.

El punto P_j está en el interior del cuadrado Q_i y P_r está en Q_s. Por lo tanto $\mathcal{F}_{i,j}$ es una fórmula elemental III o IV y $\mathcal{F}_{r,s}$, II o IV, respectivamente, (v. Figura 3.1). Esto prueba los puntos 4) y 5) del resultado.

Supongamos ahora que $\mathcal{F}_{i,j}$ no es una fórmula IV (y entonces, necesariamente es una fórmula III). Los puntos P_r y P_s están en el interior del cuadrado Q_j, pues $\mathcal{F}_{j,s}$ es una fórmula elemental IV y P_r está entre este dos puntos. De ahí que la distancia entre P_j y P_s, $\overline{P_jP_s}$ es inferior a la longitud de la semi-diagonal del cuadrado Q_j, que llamamos d_j. Análogamente, P_i y P_j están en el interior de Q_r y, por hipótesis, P_i no está en el interior del cuadrado Q_j, por lo que $d_j < d_r$. Reuniendo estas dos desigualdades, tenemos que $\overline{P_jP_s} < d_j < d_r$. Dado que $\overline{P_jP_s} = \overline{P_jP_r} + \overline{P_rP_s} < d_r$, en particular $\overline{P_rP_s} < d_r$, por lo que el punto P_s estará en el interior del cuadrado Q_r. Como también el punto P_r está en el interior de Q_s, $\mathcal{F}_{r,s}$ es una fórmula elemental IV, lo que prueba la línea 3) y termina la demostración del resultado.

□

§ 3.2.2. Estructura de las entradas no satisfacibles mínimas.— A continuación se presenta el resultado principal que nos permitirá concluir que la clase UC-4P-SAT es resoluble en tiempo polinomial. A este efecto consideraremos una instancia satisfacible minimal para deducir que cada cláusula con 4 literales contiene al menos 2 literales peliminables y que estos pueden ser identificados y eliminados mediante el algoritmo ELIMINALITERAIS. El estudio del resultado referido es extenso y se desarrolla analizando sistemáticamente las posibles estructuras del grafo de interacciones siempre que ocurre una fórmula elemental I (v. Cuadro 3.1), conteniendo una cláusula con 4 literales.

Observamos anteriormente que una instancia de la clase UC-4P-SAT que no contenga la fórmula elemental IV es satisfacible. Por tanto, cualquier instancia no satisfacible contiene, necesariamente, alguna fórmula elemental de este tipo.

Es conocido el hecho de que cualquier fórmula proposicional no satisfacible contiene una subfórmula no satisfacible minimal respecto al número de cláusulas. En el presente caso, procuramos introducir un concepto análogo de subfórmula minimal, pero preservando la estructura geométrica asociada al problema de etiquetado. En este sentido, consideremos una entrada $(\mathcal{P}, \mathcal{L})$ del problema de etiquetado UC-4P con respuesta negativa, el correspondiente grafo de interacciones $G = (V, A)$ y la correspondiente fórmula proposicional $\mathcal{F}$ (en este caso no satisfacible). Una fórmula proposicional $\mathcal{F}_1 \subset \mathcal{F}_2$ se llama minimal no satisfacible respecto al subconjunto de vértices $V_1 \subset V$ si $\mathcal{F}_1$ corresponde a la fórmula proposicional asociada al subgrafo inducido G_{V_1} y para cualquier subconjunto propio $V_2 \subsetneq V_1$ la fórmula proposicional $\mathcal{F}_2$ asociada al subgrafo inducido G_{V_2} es satisfacible.

A lo largo de la presente sección, y siempre que no se avise de lo contrario, consideraremos fórmulas proposicionales no satisfacibles mínimas en las condiciones que acabamos de definir. En el caso contrario, para una de las correspondientes entradas del problema UC-4P con respuesta negativa, siempre podríamos considerar un subconjunto con un número mínimo de puntos (vértices en el grafo de interacciones) para el cual el problema de decisión tuviese también respuesta negativa. La correspondiente fórmula proposicional es, obviamente, una subfórmula no satisfacible minimal.

Consideremos ahora el grafo de interacciones $G = (V, A)$ de una entrada no satisfacible minimal dónde $V = \{v_1, \cdots, v_t, v_{t+1}\}$. A la arista $\{v_t, v_{t+1}\} \in A(G)$ le corresponde una fórmula proposicional elemental III o IV. Designando por V_1 al subconjunto de vértices que se obtiene de V excluyendo el vértice v_{t+1}, $V_1 = V - \{v_{t+1}\}$, la fórmula proposicional $\mathcal{F}_1$ asociada al subgrafo inducido, $G(V_1)$ es satisfacible, pues V_1 es un subconjunto propio de V. Por lo tanto, existe una atribución de verdad sobre las variables asociadas a los vértices de V_1, τ^1, satisfaciendo $\mathcal{F}_1$. Pero, por construcción, ninguna de las atribuciones $\tau^1 \cup \{l_{t+1}, b_{t+1}\}$, $\tau^1 \cup \{\overline{l_{t+1}}, b_{t+1}\}$, $\tau^1 \cup \{l_{t+1}, \overline{b_{t+1}}\}$ y $\tau^1 \cup \{\overline{l_{t+1}}, \overline{b_{t+1}}\}$ podrá satisfacer $\mathcal{F}$ pues esta fórmula proposicional no es satisfacible. En estas condiciones necesariamente existe una arista $\{v_k, v_{t+1}\} \in A(G)$ asociada a una fórmula elemental III o IV de modo que la atribución τ^1 contenga los literales $\overline{l_k}, \overline{b_k}$, dónde $v_k \in V_1$. Geométricamente, este caso sucede si y sólo si el punto P_{t+1} está en el interior del cuadrado Q_k (nótese que, por construcción, $P_k < P_{t+1}$). En estas condiciones $v_k = v_t$, pues caso contrario ($k < t$), como el punto P_{t+1} está en el interior del cuadrado Q_k, el punto P_t estaría también en el interior de Q_k, ($P_k < P_t < P_{t+1}$), por lo que la fórmula elemental asociada a la arista $\{v_k, v_t\}$ sería también III o IV. Pero, en este caso, $[l_k, b_k]$ es una cláusula de $\mathcal{F}_1$, en contradicción con el hecho de que la atribución τ^1, conteniendo la atribución parcial $\{\overline{l_k}, \overline{b_k}\}$, constituye un modelo para $\mathcal{F}_1$.

Por simetría, se puede igualmente concluir que a la arista $\{v_1, v_2\}$ le corresponde una fórmula elemental II o IV.

También en las mismas condiciones, sea $G = (V, A)$ el grafo de interacciones de una entrada no satisfacible minimal. Si consideramos el subconjunto de vértices $V_2 = V_1 - \{v_t\}$ y el subgrafo inducido $G_2 = G(V_2)$, se verifica que para cualquier modelo para la fórmula proposicional asociada $\mathcal{F}_2$ las posiciones de las etiquetas de los puntos $P_1, \cdots, P_{t-1}$ bloquean todas las posiciones para el posicionamiento de la etiqueta L_t excepto la posición 1, Q_t^1 (v. Figura 3.1). Para que tal cosa ocurra, deben existir forzosamente dos etiquetas L_α y L_β, con $P_\alpha < P_t$ y $P_\beta < P_t$ tal que una de ellas bloquee la posición Q_t^2 y la otra, la posición Q_t^4. En las condiciones del problema, cualquiera de las dos etiquetas, L_α y L_β, bloquea (también) la posición Q_t^3. En estas condiciones la etiqueta L_t solamente puede ocupar la posición Q_t^1 sin que haya superposición con las etiquetas L_α y L_β. I.e., Q_t^1 es una posición forzada para la etiqueta L_t.

Considerando un par de puntos P_i y P_j y sus correspondientes etiquetas L_i y L_j, respectivamente, observemos las condiciones en que puede haber bloqueos al posicionamiento de etiquetas sin que haya falsificación de la entrada:

B1 - Posición Q_j^2 bloqueada. El punto P_j deberá estar en el interior del cuadrado Q_i, el punto P_i en el interior del cuadrado Q_j y la etiqueta L_i ocupar la posición Q_i^2. Por tanto, $\mathcal{F}_{i,j}$ es una fórmula elemental IV y los literales l_i y $\overline{b_i}$ deberán estar atribuidos (v. B1, Cuadro 3.2).

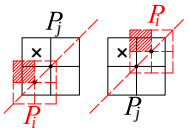
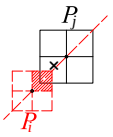
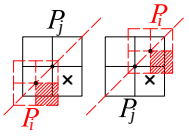
B2 - Posición Q_j^3 bloqueada. Este bloqueo sucede en dos casos: independientemente de la posición de la etiqueta L_i, cuando P_i está en el interior del cuadrado Q_j, lo que corresponde a que $\mathcal{F}_{i,j}$ es una fórmula elemental II o IV; o cuando P_i es exterior al referido cuadrado, $\mathcal{F}_{i,j}$ es una fórmula elemental I y los literales $\overline{l_i}, \overline{b_i}$ atribuidos (v. B2, Cuadro 3.2).

B3 - Posición Q_j^4 bloqueada. Sucede para una configuración simétrica a B1, i. e., a la arista $\{v_i, v_j\}$ le corresponde la fórmula elemental IV y los literales $\overline{l_i}, b_i$ atribuidos (v. B3, Cuadro 3.2).

El bloqueo de la posición Q_j^1 sólo sucede si existe un punto $P_r > P_j$ de modo que $\mathcal{F}_{j,r}$ es una fórmula elemental de tipo III o IV. Puede ser incluso del tipo I si $\{l_r, b_r\} \subset \tau$. Si $P_r < P_j$ y la posición Q_j^1 estuviese bloqueada por la posición de la etiqueta L_r entonces todas las demás posiciones de la etiqueta L_j: Q_j^2, Q_j^3 y Q_j^4, estarían también bloqueadas.

Con el objetivo de caracterizar el grafo de interacciones de una instancia minimal no satisfacible, consideremos $\mathcal{F}$ la fórmula proposicio-

Cuadro 3.2: Bloqueos al posicionamiento de etiquetas.

B1		Posición Q_j^2 bloqueada $\{v_i, v_j\}$ arista IV y $l_i, \overline{b_i}$ atribuidos
B2		Posición Q_j^3 bloqueada $\{v_i, v_j\}$ arista I ($v_i < v_j$) y $\overline{l_i}, \overline{b_i}$ atribuidos o $\{v_i, v_j\}$ arista II o IV
B3		Posición Q_j^4 bloqueada $\{v_i, v_j\}$ arista IV y $\overline{l_i}, b_i$ atribuidos

nal de una entrada del problema UC-4P. Sea $G = (V, A)$ el correspondiente grafo de interacciones y $G_1 = G(V_1)$ el subgrafo inducido por $V_1 = V - \{v_{t+1}\}$. Como observamos anteriormente, cualquier modelo para $\mathcal{F}_1$ contiene la atribución $\{\overline{l_t}, \overline{b_t}\}$.

De un modo general supongamos que v_k es uno de los vértices de V_1 para el cual $\{\overline{l_k}, \overline{b_k}\} \subset \tau^1$ para cualquier modelo de $\mathcal{F}_1$. Entonces, la etiqueta forzosamente ocupa la posición Q_k^1. Esto sólo podrá ocurrir si existen dos puntos P_α y P_β, menores que el punto P_k, cuyas etiquetas, ya posicionadas por la referida atribución, bloqueen a las posiciones Q_k^2, Q_k^3 y Q_k^4. A este efecto, se tiene necesariamente que $\mathcal{F}_{\alpha,k}$ y $\mathcal{F}_{\beta,k}$ son fórmulas elementales IV y las etiquetas L_α y L_β ocupan las posiciones correspondientes a la atribución $\{l_\alpha, \overline{b_\alpha}, \overline{l_\beta}, b_\beta\}$ o $\{\overline{l_\alpha}, b_\alpha, l_\beta, \overline{b_\beta}\}$.

Como los puntos de V están ordenados, podemos suponer a lo largo de la presente sección que $P_\alpha < P_\beta < P_t$.

Consideremos una atribución de verdad parcial τ y supongamos, sin pérdida de generalidad, que $\{l_\alpha, \overline{b_\alpha}, \overline{l_\beta}, b_\beta\} \subset \tau$. Para que $\{\overline{l_k}, \overline{b_k}\}$ sea forzada las atribuciones de verdad $\{l_\alpha, \overline{b_\alpha}\}$ y $\{\overline{l_\beta}, b_\beta\}$ también deben ser forzadas. A estas condiciones corresponde geométricamente el que la etiqueta L_α ocupe la posición Q_α^2 y la etiqueta L_β ocupa la posición Q_β^4 bloqueando a las posiciones Q_k^2 y Q_k^4 de la etiqueta L_k, respectivamente. La posición Q_k^3 está bloqueada pues los puntos P_α y P_β están ambos en el interior del cuadrado Q_k^3. Consideremos que v_α y v_β son los vértices mayores en estas condiciones.

Dado que el punto P_β está en el interior del cuadrado Q_α y admitiendo que el punto P_α también está en el interior del cuadrado Q_β (y como por hipótesis $P_\alpha < P_\beta$, este estaría en el interior del cuadrado Q_β^3), a la arista $\{v_\alpha, v_\beta\}$ le corresponde una fórmula elemental IV. Como por hipótesis la etiqueta L_α ocupa la posición Q_α^2 ($\{l_\alpha, \overline{b_\alpha}\} \subset \tau$) y L_β, la posición

Q^4_β ($\{\overline{l_\beta}, b_\beta\} \subset \tau$), la posición Q^1_k de la etiqueta L_k es obligatoria. Forzada por las posiciones de las etiquetas L_α y L_β si y sólo si la posición Q^3_α de la etiqueta L_α está bloqueada, pues la posición Q^4_α queda bloqueada por la etiqueta L_β. Por tanto deberá existir un vértice $v_\delta < v_\alpha$ de modo que a la arista $\{v_\delta, v_\alpha\}$ le corresponda una fórmula elemental I, II o IV. Si $\mathcal{F}_{\delta,\alpha}$ es una fórmula elemental I es también necesario que $\{\overline{l_\delta}, \overline{b_\delta}\} \subset \tau$ sea una atribución forzada. Es decir, constituye condición suficiente para que la posición Q^1_k de la etiqueta L_k esté forzada que G contenga el grafo de la izquierda en la Figura 3.8 como subgrafo y también que a la arista $\{v_\delta, v_\alpha\}$ le corresponda la fórmula elemental II o IV o bien la fórmula I con la posición Q^1_δ de la etiqueta L_δ forzada.

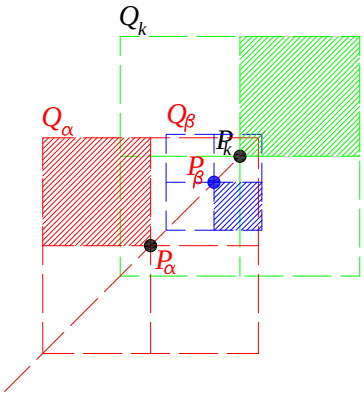


Figura 3.7: Para que la etiqueta L_k sea forzada a ocupar la posición Q^1_k, las posiciones 2, 3 y 4 del cuadrado Q_k deberán estar bloqueadas.

Caso contrario, si P_α no está en el interior del cuadrado Q_β, entonces a la arista $\{v_\alpha, v_\beta\}$ le corresponde una fórmula elemental III (v. Figura 3.7) por lo que G contiene al grafo de la derecha en la Figura 3.8 como subgrafo. Para que la posición Q^2_β de la etiqueta L_β esté bloqueada deberá existir otro vértice, $v_{i_1} \in V_1$, de modo que la etiqueta L_{i_1} ocupe la posición $Q^2_{i_1}$, bloqueando aquella posición. Para que esto ocurra, $\mathcal{F}_{\beta,i_1}$ deberá ser una fórmula elemental IV y $\{l_{i_1}, \overline{b_{i_1}}\} \subset \tau$. Por lo que el vértice v_{i_1} podrá ocupar una de dos posiciones respecto al vértice v_β, $v_{i_1} < v_\beta$ (A.1) o $v_\beta < v_{i_1}$ (A.2).

A.1. $(v_\alpha <) v_{i_1} < v_\beta$**.** El vértice v_{i_1} es mayor que v_α, pues en caso contrario a las aristas $\{v_{i_1}, v_\alpha\}$ y $\{v_\alpha, v_k\}$ estarían en las condiciones del Lema 3.4, por lo que $\mathcal{F}_{\alpha,\beta}$ sería una fórmula elemental IV, en contradicción con la hipótesis.

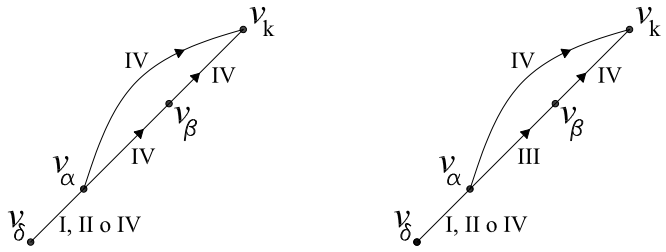


Figura 3.8: Grafo de interacciones con cuatro vértices (v. texto).

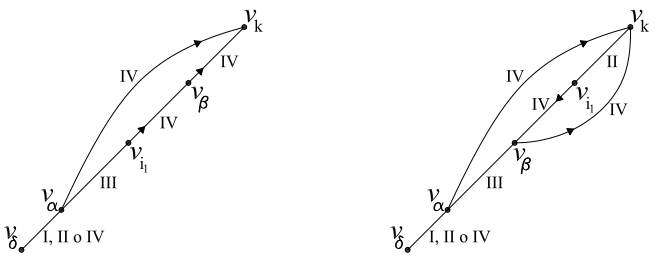
Dado que el punto P_{i_1} está entre los puntos P_α y P_β ($v_\alpha < v_{i_1} < v_\beta$), P_{i_1} está en el interior del primer cuadrante de Q_α. Por otro lado, como $\{l_{i_1}, \overline{b_{i_1}}, l_\alpha, \overline{b_\alpha}\} \subset \tau$, a la arista $\{v_\alpha, v_{i_1}\}$ no le puede corresponder la fórmula elemental IV, (si así fuera, se tendría $\mathcal{F}_{\alpha,i_1}(\tau) = 0$), por lo que a la arista $\{v_\alpha, v_{i_1}\}$ le corresponde una fórmula elemental III (v. Figura 3.9-(A.1)).

A.2. $v_\beta < v_{i_1} (< v_k)$**.** El vértice $v_{i_1} < v_k$, pues en caso contrario, (si fuese $v_{i_1} > v_k$) como $\{v_\alpha, v_k\}$ y $\{v_\beta, v_{i_1}\}$ estarían en las condiciones del Lema 3.4 y $\mathcal{F}_{k,i_1}$ sería una fórmula elemental IV. Como la posición Q_k^1 no está bloqueada (por hipótesis) P_{i_1} no podría estar en el interior de Q_k^1 por lo que la longitud de la diagonal de la etiqueta L_β excedería la longitud de la diagonal de la etiqueta L_k, $d_\beta > d_k$. Cómo $P_\alpha < P_\beta < P_k$ y $\mathcal{F}_{\alpha,k}$ es una fórmula elemental IV, el punto P_α estaría en el interior del cuadrado Q_β por lo que a la arista $\{v_\alpha, v_\beta\}$ le correspondería una fórmula IV, en contradicción con la hipótesis.

Suponiendo que $v_\beta < v_{i_1} < v_k$, a la arista $\{v_{i_1}, v_k\}$ le corresponde la fórmula proposicional II, pues el punto P_{i_1} está en el interior del tercer cuadrante del cuadrado Q_k y no puede bloquear a Q_k^2, pues, por hipótesis, v_α es el mayor vértice cuya etiqueta bloquea esta posición.

En este caso para que la posición Q_α^3 esté bloqueada es igualmente necesario que exista un vértice, digamos $v_\delta < v_\alpha$ de modo que a la arista $\{v_\delta, v_\alpha\}$ le corresponda una fórmula elemental I, II o IV.

La Figura 3.9 presenta los grafos de interacciones relativos a los casos analizados, A.1 y A.2. En cualquiera de ellos, la posición $Q_{i_1}^4$ tendrá que estar bloqueada. Por tanto, existe otro vértice, $v_{i_2} \in V_1$ de modo que la posición de la correspondiente etiqueta, posicionada de acuerdo a la atribución de verdad τ, bloquee a la posición 4 de la etiqueta L_{i_1}. Para que estas condiciones ocurran, $\mathcal{F}_{i_1,i_2}$ deberá ser una fórmula elemental IV y la atribución $\{\overline{l_{i_2}}, b_{i_2}\} \subset \tau$ i.e. L_{i_2} debería ocupar la posición $Q_{i_2}^4$.



(A.1) **(A.2)**

Figura 3.9: Grafo de interacciones con cinco vértices (ver texto, casos A.1 y A.2.

Para analizar todos los casos posibles, empezaremos considerando, a partir de A.1, los casos $v_{i_2} < v_{i_1}$ (A.1.1) y $v_{i_1} < v_{i_2}$ (A.1.2).

A.1.1. $(v_\alpha <) v_{i_2} < v_{i_1} < v_\beta < v_k$**.** Comencemos por observar que $v_\alpha < v_{i_2}$, pues en caso contrario las aristas $\{v_\alpha, v_k\}$ y $\{v_{i_2}, v_{i_1}\}$ estarían en las condiciones del Lema 3.4, por lo que a la arista $\{v_\alpha, v_{i_1}\}$ le correspondería una fórmula elemental IV, en contradicción con la hipótesis.

Considerando que $v_\alpha < v_{i_2} < v_{i_1} < v_\beta < v_k$ y que $\mathcal{F}_{\alpha,k}$ es una fórmula elemental IV, el punto P_k está en el interior del cuadrado Q_α, por lo que P_{i_2} también está en Q_α e, por consiguiente, a la arista $\{v_\alpha, v_{i_2}\}$ le corresponde una fórmula III o IV.

A.1.2. $v_\alpha < v_{i_1} < v_{i_2} (< v_\beta)$**.** Suponiendo $v_{i_1} < v_{i_2}$, se tiene que $v_{i_2} < v_\beta$. En caso contrario, la secuencia de vértices $v_\alpha < v_{i_1} < v_\beta < v_{i_2}$ estaría en las condiciones del Lema 3.4 por lo que $\mathcal{F}_{\alpha,i_1}$ o $\mathcal{F}_{\beta,i_2}$ sería una fórmula elemental IV. Como, por hipótesis, $\mathcal{F}_{\alpha,i_1}$ es una fórmula de tipo III, $\mathcal{F}_{\beta,i_2}$ tendría que ser una fórmula IV. Pero, dado que $\{\overline{l_\beta}, b_\beta, \overline{l_{i_2}}, b_{i_2}\} \subset \tau$, se tiene que $\mathcal{F}_{\beta,i_2}(\tau) = 0$, en contradicción con la hipótesis de que τ constituya un modelo para $\mathcal{F}_1$.

Dado que $v_{i_1} < v_{i_2} < v_\beta$ y que $\mathcal{F}_{i_1,\beta}$ es una fórmula IV, los puntos P_{i_1} y P_{i_2} están en el interior del cuadrado Q_β, por lo que $\mathcal{F}_{i_2,\beta}$ es una fórmula II o IV. Como no puede ser IV (pues, dado que $\{\overline{l_\beta}, b_\beta, \overline{l_{i_2}}, b_{i_2}\} \subset \tau$, $\mathcal{F}_{\beta,i_2}(\tau) = 0$, existiría una contradicción con la hipótesis de que τ sea un modelo para $\mathcal{F}_1$) se tiene que $\mathcal{F}_{i_2,\beta}$ es una fórmula elemental II.

A partir de A.2 y considerando a las dos posibles posiciones relativas del vértice v_{i_2}, $v_{i_2} < v_{i_1}$ (A.2.1) y $v_{i_1} < v_{i_2}$ (A.2.2) se tiene:

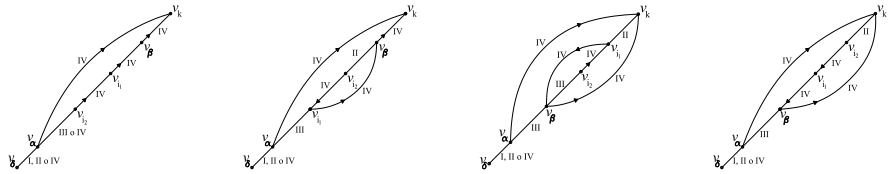
A.2.1. $(v_\beta <) v_{i_2} < v_{i_1} < v_k$**.** En este caso, se tiene necesariamente que $v_\beta < v_{i_2}$. En caso contrario, $v_{i_2} < v_\beta < v_{i_1} < v_k$ y, de acuerdo

con el Lema 3.4, $\mathcal{F}_{i_2,\beta}$ sería una fórmula elemental IV, ya que, por hipótesis, $\mathcal{F}_{i_1,k}$ es una fórmula II. Lo que es absurdo pues, en este caso, $\{\overline{l_\beta}, b_\beta, \overline{l_{i_2}}, b_{i_2}\} \subset \tau$ y, por tanto, $\mathcal{F}_{\beta,i_2}(\tau) = 0$, en contradicción con la hipótesis de que τ sea un modelo para $\mathcal{F}_1$.

Dado que $v_\beta < v_{i_2} < v_{i_1}$ y $\mathcal{F}_{\beta,i_1}$ es una fórmula elemental IV, el punto P_{i_2} está en el interior del cuadrado Q_β. Sin embargo, por el Lema 3.2, a la arista $\{v_\beta, v_{i_2}\}$ sólo le puede corresponder una fórmula elemental III.

A.2.2. $v_\beta < v_{i_1} < v_{i_2} (< v_k)$. Suponiendo, por reducción al absurdo, que $v_{i_2} > v_k$ y dado que $v_\beta < v_{i_1} < v_k < v_{i_2}$, por el Lema 3.4, $\mathcal{F}_{i_1,k}$ es una fórmula IV, en contradicción con la hipótesis. Se deduce, por tanto, que $v_{i_2} < v_k$.

Por otro lado, considerando que $v_\beta < v_{i_2} < v_k$ y que $\mathcal{F}_{\beta,k}$ es una fórmula IV, el punto P_{i_2} está en el interior del cuadrado Q_k y $\{\overline{l_{i_2}}, b_{i_2}\} \subset \tau$. Dado que v_β es el mayor vértice cuya etiqueta bloquea la posición Q_k^4, $\mathcal{F}_{i_2,k}$ no puede ser una fórmula IV y sólo puede ser una fórmula elemental II.



(A.1.1) **(A.1.2)** **(A.2.1)** **(A.2.2)**

Figura 3.10: Grafos de interacciones con seis vértices (v. texto, casos A.1.1, A.1.2, A.2.1 y A.2.2).

En todas las configuraciones analizadas, A.1.1, A.1.2, A.2.1 y A.2.2, excepto en A.1.1 se verifica que cuando $\mathcal{F}_{\alpha,i_2}$ es una fórmula elemental IV, para que la posición $Q_{i_2}^4$ esté forzada es necesario que la posición $Q_{i_2}^2$ esté bloqueada. Este hecho implica la existencia de un vértice $v_{i_3} \notin \{v_\alpha, v_\beta, v_{i_1}, v_{i_2}, v_k\}$ de modo que $\{l_{i_3}, \overline{b_{i_3}}\} \in \tau$.

En A.1.1, en el caso de que $\mathcal{F}_{\alpha,i_2}$ sea una fórmula IV, a cada arista definida por vértices consecutivos del ciclo, $C_2 = \{v_\alpha, v_k, v_\beta, v_{i_1}, v_{i_2}, v_\alpha\}$ le corresponde una fórmula elemental IV. En estas condiciones, siguiendo el Lema 3.3, $\{\overline{l_k}, \overline{b_k}\} \subset \tau$ si y sólo si la posición Q_α^3 está bloqueada. Lo que obviamente sucede cuando $\mathcal{F}_{\delta,\alpha}$ es una fórmula II o IV o incluso I, en el caso de que $\{\overline{l_\delta}, \overline{b_\delta}\}$ sea una atribución forzada.

Si consideramos el vértice v_β en las condiciones anteriores ($v_\alpha < v_\beta < v_k$) y de acuerdo a los casos A.1 y A.2, existe un vértice v_{i_1} que

podrá ocupar una de las posiciones relativas $v_\beta < v_{i_1} (< v_k)$ o $(v_\alpha <)v_{i_1} < v_\beta$. Para representar estas dos posibilidades consideraremos un árbol binario cuya raíz corresponde al vértice v_β. En el primer nivel (vértices adyacentes a v_β), cada uno de los (dos) vértices representa una de las dos posiciones relativas, sobre v_β y bajo v_β (v. Figura 3.11). De acuerdo con los casos estudiados en A.1.1, A.1.2, A.2.1 y A.2.2, el grafo de interacciones contiene otro vértice v_{i_2} ocupando una de las cuatro posiciones relativas siguientes $v_\beta < v_{i_1} < v_{i_2}$, $v_\beta < v_{i_2} < v_{i_1}$, $v_{i_1} < v_{i_2} < v_\beta$ y $v_{i_2} < v_{i_1} < v_\beta$. Los cuatro vértices en el segundo nivel representan las posiciones relativas de este vértice (v_{i_2}).

Puede ocurrir uno de dos casos. O bien el grafo de la izquierda (A.1.1) en la Figura 3.10 es un subgrafo del grafo de interacciones G y a la arista $\{v_{i_2}, v_\alpha\}$ le corresponde una fórmula elemental IV o bien, en caso contrario, existen dos vértices v_{i_3} y v_{i_4} de modo que $\mathcal{F}_{i_3,i_4}$ es una fórmula IV y que las etiquetas L_{i_3} y L_{i_4} ocupan las posiciones $Q^2_{i_3}$ y $Q^4_{i_4}$, respectivamente. El vértice v_{i_3} podrá ocupar una de las ocho posiciones relativas posibles correspondientes al tercer nivel en el árbol binario (v. Figura 3.11). Cuatro de estas posiciones se dan para $v_\beta < v_{i_1}$ y las otras cuatro para $v_{i_1} < v_\beta$. Análogamente se puede concluir que v_{i_4} podrá ocupar una de 16 posiciones .

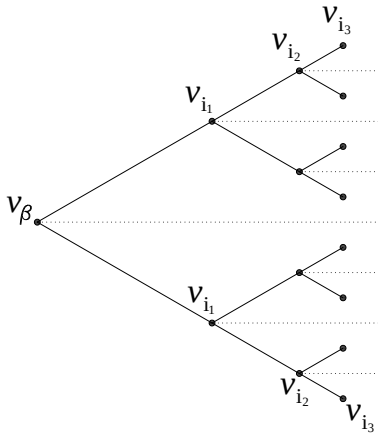


Figura 3.11: Árbol binario. Los vértices del nivel k representan las posibles posiciones de cada un de los vértices del camino W_m.

Por inducción, consideremos el camino $W_m = \{v_\beta, v_{i_1}, \cdots, v_{i_m}\}$, con $m \geq 2$ par, el correspondiente árbol binario construido en las condiciones anteriores y la atribución de verdad $\tau_{W_m} = \{\overline{l_\beta}, b_\beta, l_{i_1}, \overline{b_{i_1}}, \cdots, \overline{l_{i_m}}, b_{i_m}\}$.

En estas condiciones podrá ocurrir una de dos situaciones. O a la arista $\{v_{i_m}, v_\alpha\}$ le corresponde una fórmula elemental IV, y en este caso el ciclo $C_m = \{v_\alpha, v_k, v_\beta, v_{i_1}, \ldots, v_{i_m}, v_\alpha\}$ está en las condiciones del Le-

ma 3.3 y, por tanto, la atribución $\{\overline{l_k}, \overline{b_k}\}$ es forzada si y sólo si la posición Q^3_α está bloqueada. O bien, en el caso contrario, $\mathcal{F}_{i_m,\alpha}$ es una fórmula III por lo que existe un vértice $v_{i_{m+1}}$ de modo que a la arista $\{v_{i_m}, v_{i_{m+1}}\}$ le corresponda una fórmula elemental IV y $\{l_{i_{m+1}}, \overline{b_{i_{m+1}}}\} \subset \tau$.

En este último caso, considerando los tres últimos vértices del camino W_m, $v_{i_{m-2}}, v_{i_{m-1}}, v_{i_m}$, existen cuatro posibles posiciones relativas, representadas en la Figura 3.12, $v_{i_{m-2}} < v_{i_{m-1}} < v_{i_m}$ (C.1), $v_{i_{m-2}} < v_{i_m} < v_{i_{m-1}}$ (C.2), $v_{i_{m-1}} < v_{i_m} < v_{i_{m-2}}$ (C.3) y $v_{i_m} < v_{i_{m-1}} < v_{i_{m-2}}$ (C.4).

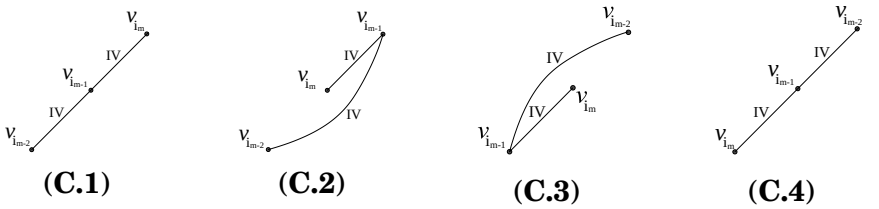


Figura 3.12: Posibles caminos simples de longitud 2 en un grafo de interacciones de una instancia de UC-4P (v. texto).

Para cada una de las configuraciones referidas anteriormente: C.1, C.2, C.3 y C.4, existen cuatro distintas posiciones posibles para el vértice $v_{i_{m+1}}$ lo que totaliza dieciséis configuraciones diferentes, presentadas en la Figura 3.13.

Analizando todas ellas se observa que la mitad de las configuraciones no puede ocurrir. Si se satisficieran las condiciones C.1.3, C.2.3, C.3.3, C.3.4 y C.4.3, de acuerdo con la línea 1) del Lema 3.4 resultaría una contradicción con la hipótesis de $\{\overline{l_{i_{m-2}}}, b_{i_{m-2}}, l_{i_{m-1}}, \overline{b_{i_{m-1}}}, \overline{l_{i_m}}, b_{i_m}, l_{i_{m+1}}, \overline{b_{i_{m-2}}}\} \subset \tau$ y esta asignación constituir un modelo para $\mathcal{F}_1$. En la misma línea de argumentación, las configuraciones no pueden ocurrir pues, en caso contrario, en consecuencia de la línea 3) del mismo Lema 3.4, τ no podría ser un modelo para $\mathcal{F}_1$.

Tampoco las configuraciones C.1.4 o C.4.4 pueden darse. Consideremos, por absurdo, y, sin perdida de generalidad, que C.1.4 sucede. Para la configuración C.4.4 el razonamiento seria idéntico, considerando la simetría entre C.1.4 y C.4.4. El vértice v_β no puede estar entre v_{i_m} y $v_{i_{m+1}}$. Si lo estuviera, como a la arista $\{v_\beta, v_k\}$ le corresponde una fórmula IV, de acuerdo con el Lema 3.4, $\mathcal{F}_{\beta,i_m}$ sería una fórmula IV. Además, a cada una de las aristas del subgrafo completo inducido $G(\{v_{i_{m+1}}, v_\beta, v_{i_m}\})$, le corresponde una fórmula IV. (Se recuerde que el vértice v_β había sido tomado como el mayor vértice cuya fórmula es del tipo IV, por lo tanto $\mathcal{F}_{k,i_m}$ el una fórmula II.) Entonces, por el Lema 3.1, se tiene $\{l_{i_{m+1}}, b_{i_{m+1}}\} \subset \tau$ o $\{\overline{l_{i_m}}, \overline{b_{i_{m+1}}}\} \subset \tau$, lo que contradice la hipótesis. Como v_β no puede estas entre v_{i_m} y $v_{i_{m+1}}$, entonces o $v_\beta < v_{i_{m+1}}$ o $v_\beta > v_{i_m}$.

En cualquier caso, como existe un camino uniendo los vértices v_β y $v_{i_{m-2}}$ (y $v_{i_{m+1}} < v_{i_{m-2}} < v_{i_m}$), existe una arista $\{v_{i_j}, v_{i_{j+1}}\}$ de modo que esta arista y $\{v_{i_m}, v_{i_{m+1}}\}$ están en las condiciones del Lema 3.4. Por lo tanto existe un subgrafo completo K_3 conteniendo tres vértices del camino W_{m+1} en las condiciones del Lema 3.1. Lo que está en contradicción con la hipótesis pues τ_{W_m} no contiene, para un mismo vértice, dos literales ambos afirmados o ambos negados. Por lo tanto, en el presente caso esto no puede ocurrir.

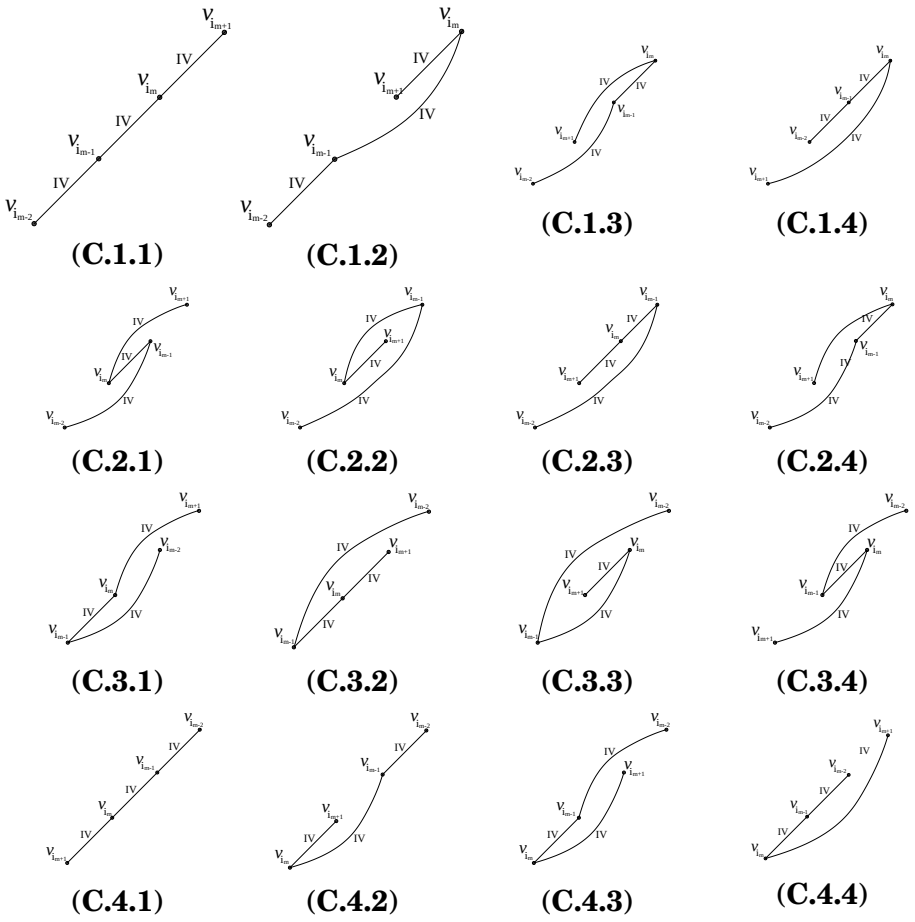


Figura 3.13: Análisis de las configuraciones para un nuevo vértice, $v_{i_{m+1}}$.

Por lo que los casos que quedan, C.1.1, C.1.2, C.2.2, C.2.3, C.3.2, C.3.3, C.4.1 y C.4.2, corresponden a los ocho caminos de cada rama, con raíz en $v_{i_{m-2}}$.

Para el camino $W_{m+1} = \{v_\beta, v_{i_1}, \ldots, v_{i_m}, v_{i_{m+1}}\}$, $v_{i_j} > v_{i_{j+1}}$ se tiene que para cada $i_j = i_\beta, i_1, \cdots, i_m$, a la arista $\{v_{i_{m+1}}, v_\alpha\}$ no le puede co-

rresponder una fórmula IV. Si así fuera, como $\{l_\alpha, \overline{b_\alpha}, l_{i_{m+1}}, \overline{b_{i_{m+1}}}\} \subset \tau$, $\mathcal{F}_{\alpha,i_{m+1}}(\tau) = 0$, en contradicción con la hipótesis. Por lo tanto, existe un vértice $v_{i_{m+2}} \in V_1$ de modo que a la arista $\{v_{i_{m+1}}, v_{i_{m+2}}\}$ le corresponde una fórmula IV y $\{\overline{l_{i_{m+2}}}, b_{i_{m+2}}\} \subset \tau$.

Nuevamente, considerando los tres últimos vértices de la secuencia W_{m+1}, la ampliación sigue las mismas condiciones que vimos anteriormente. Por tanto, o existe un ciclo conteniendo el camino W_{m+2}, $C_{m+2} = \{v_\alpha, v_k, v_\beta, v_{i_1}, \ldots, v_{i_m}, v_{i_{m+1}}, v_{i_{m+2}}, v_\alpha\}$ para lo cual a la arista $\{v_{i_{m+2}}, v_\alpha\}$ le corresponde una fórmula IV o, en caso contrario, dado que $m + 2$ es par, podemos repetir los dos pasos anteriores.

Dado que el número de vértices es finito, en algún paso necesariamente se obtiene un ciclo en las condiciones referidas, por lo que:

Lema 3.5. *Sean $G = (V, A)$ el grafo de interacciones de una instancia no satisfacible minimal, $V = \{v_1, \ldots, v_t, v_{t+1}\}$, el conjunto de vértices y $\mathcal{F}$ la correspondiente fórmula proposicional. Si $V_1 = V - \{v_{t+1}\}$, $G_1 = G(V_1)$ el subgrafo inducido por V_1 y τ un modelo para la correspondiente fórmula asociada, $\mathcal{F}_1$, entonces, para cada atribución forzada $\{\overline{l_k}, \overline{b_k}\}$, se verifican:*

a) Existe un ciclo $C_m = \{v_\alpha, v_k, v_\beta, v_{i_1}, \cdots, v_{i_m}, v_\alpha\}$, con m par, para el cual a cada par de vértices consecutivos le corresponde una arista asociada a una fórmula elemental IV.

b) Existe una arista $\{v_\delta, v_\alpha\}$, con $v_\delta < v_\alpha$, de modo que $\mathcal{F}_{\delta,\alpha}$ es una fórmula II *o* IV *o* I *y, en este último caso, $\{\overline{l_\delta}, \overline{b_\delta}\}$ es una atribución forzada.*

Como consecuencia del Lema 3.5, se puede concluir que G es un grafo en las condiciones de la Figura 3.14.

§ 3.2.3. Satisfacibilidad de la clase UC-4P-SAT.— Consideremos una instancia $\mathcal{F}$ del problema UC-4P-SAT, no satisfacible que, sin pérdida de generalidad, podemos suponer no satisfacible minimal. Designamos por $G = (V, A)$ al respectivo grafo de interacciones, $V = \{v_1, \cdots, v_t, v_{t+1}\}$ al conjunto de los vértices y $\mathcal{F}_1$ a la fórmula proposicional satisfacible asociada al subgrafo inducido $G_1 = G(V - \{v_{t+1}\})$.

Dado que cualquier modelo τ de $\mathcal{F}_1$ contiene la atribución $\{\overline{l_t}, \overline{b_t}\}$, existe un ciclo $C_m = \{v_\alpha, v_t, v_\beta, v_{i_1}, \cdots, v_{i_m}, v_\alpha\}$, por el Lema 3.5. Por otro lado, $\mathcal{F}_{t,t+1}$ es una fórmula elemental III o IV por lo que, de acuerdo al Lema 3.3, la propagación unitaria de la atribución $\{\overline{l_\alpha}\}$ y de la atribución $\{\overline{b_\alpha}\}$ falsifican la fórmula proposicional asociada al subgrafo inducido $G(C_m \cup \{v_{t+1}\})$.

Pero, en el caso de que a la arista $\{v_\delta, v_\alpha\}$ le corresponda una fórmula elemental I entonces la etiqueta L_δ forzosamente ocupa la posición Q^1_δ.

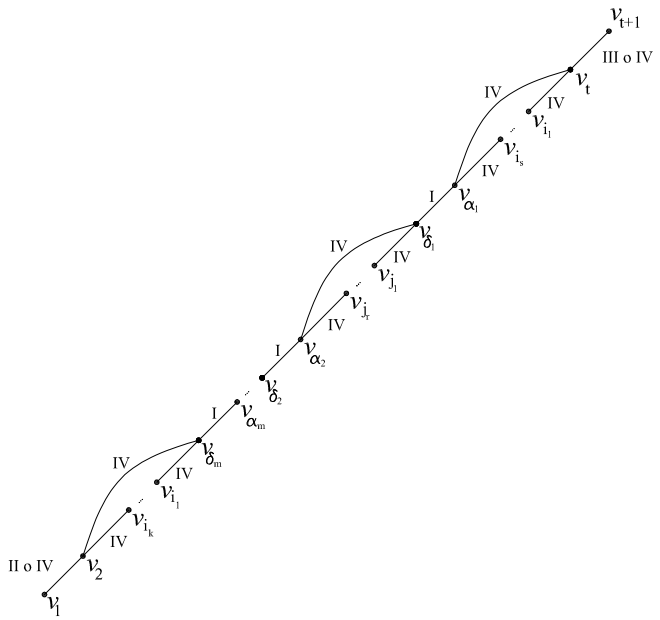


Figura 3.14: Estructura de un grafo de interacciones de una entrada minimal con respuesta negativa.

Así, $\mathcal{F}_{\delta,\alpha} = \{[l_\delta, b_\delta, \overline{l_\alpha}, \overline{b_\alpha}]\} \subset \mathcal{F}$ y sus literales $\overline{l_\alpha}$ y $\overline{b_\alpha}$ son p-eliminables (en $\mathcal{F}$, independientemente del orden de exclusión) pues $\{\overline{l_\alpha}\} \Rrightarrow 0$ y $\{\overline{b_\alpha}\} \Rrightarrow 0$. Excluyéndolos se obtiene la cláusula $[l_\delta, b_\delta]$. Por lo que, por cada ciclo en las condiciones del Lema 3.5 puede repetirse nuevamente el argumento anterior.

Considerando el algoritmo ELIMINALITERALES($\mathcal{F}$), durante su procesamiento, en algún paso, se produce la cláusula nula. Por reducción al absurdo, supongamos que la fórmula $\mathcal{F}^E =$ ELIMINALITERALES($\mathcal{F}$) no contiene la cláusula nula. Tampoco puede ser una instancia 2SAT, pues se tendría una fórmula 2SAT sin literales p-eliminables y sin la cláusula nula, por tanto satisfacible, contradiciendo la hipótesis. Por lo tanto existe una cláusula $D' \subset [l_\delta, b_\delta, \overline{l_\alpha}, \overline{b_\alpha}]$ con 3 o 4 literales.

Supongamos que v_α es el mayor vértice de G en estas condiciones. De acuerdo con el Lema 3.5, existe un ciclo $C = \{v_\alpha, v_k, v_\beta, v_{i_1}, \cdots, v_{i_m}, v_\alpha\}$ para lo cual $\{\overline{l_\alpha}\} \Rrightarrow \{\overline{l_k}, \overline{b_k}\}$ y $\{\overline{b_\alpha}\} \Rrightarrow \{\overline{l_k}, \overline{b_k}\}$ en $\mathcal{F}$. Y, por supuesto en $\mathcal{F}^E$. Como, además, la subfórmula $\mathcal{F}^E_C$ contiene una subcláusula de $[l_k, b_k]$ y $\overline{l_\alpha} \in D'$ o $\overline{b_\alpha} \in D'$, D' contiene un literal p-eliminable. Lo que contradice la hipótesis pues $\mathcal{F}^E$ había sido tomada sin literales p-eliminables. Aunque la fórmula $\mathcal{F}$ no sea minimal, si no es satisfacible, la fórmula obtenida por el algoritmo ELIMINALITERALES($\mathcal{F}$) sigue conteniendo la cláusula nula.

Se concluye, por tanto, que la fórmula proposicional devuelta por el algoritmo ELIMINALITERALES($\mathcal{F}$) contiene la cláusula nula si y sólo si $\mathcal{F}$ es una instancia de UC-4P-SAT no satisfacible, para cualquiera ordenación de sus cláusulas y literales.

Por lo tanto, podemos enunciar el siguiente resultado:

Teorema 3.6. *Sea $\mathcal{F}$ una instancia del problema* UC-4P-SAT. *$\mathcal{F}$ es satisfacible si y sólo si la fórmula proposicional equivalente obtenida mediante el algoritmo* ELIMINALITERALES($\mathcal{F}$) *no contiene la cláusula nula.*

Demostración. Sea $\mathcal{F}$ una instancia de UC-4P-SAT no satisfacible. Entonces, como hemos visto, para cualquiera subfórmula minimal no satisfacible $\mathcal{F}^0 \subset \mathcal{F}$, el algoritmo ELIMINALITERALES($\mathcal{F}^0$) devuelve una formula con la cláusula nula. Por lo tanto, lo mismo ocurre con ELIMINALITERALES($\mathcal{F}$). □

Y, como consecuencia,

Teorema 3.7. *La clase de satisfacibilidad* UC-4P-SAT *es una subclase (propia) de la clase de satisfacibilidad polinomial* PURL.

Demostración. Sea $\mathcal{F}$ una instancia de UC-4P-SAT. Como para cualquiera ordenación de sus cláusulas y literales $\mathcal{F}$ es satisfacible si y sólo si la fórmula lógicamente equivalente $\mathcal{F}^E$=ELIMINALITERALES($\mathcal{F}$) no contiene la cláusula nula, de acuerdo con la Definición 2.3, UC-4P-SAT es subclase de PURL. □

Como conclusión final, se obtiene el siguiente resultado:

Corolario 3.8. *El problema de etiquetado* UC-4P *es resoluble en tiempo polinomial.*

Demostración. Dado que cada instancia del problema de etiquetado UC-4P se puede reducir a una CNF-fórmula en tiempo cuadrático relativamente al numero de puntos a etiquetar, el resultado es una consecuencia directa del Teorema 3.7 y del Teorema 2.6. □

CAPÍTULO 4

Emparejamiento ortogonal simple en el cilindro
EOSC

La representación gráfica de grafos se puede considerar como una forma de visualización de datos con un amplio campo de aplicaciones tales como diseño de circuitos electrónicos, genealogía, sociología, programación, bases de datos y cartografía. Algunos de los aspectos a considerar para la representación de un grafo son: el tipo de aristas: rectilíneas, rectilíneas con cambios de dirección (codos) o curvilíneas; el número de cruces entre aristas; el alejamiento mínimo entre vértices o aristas; el área ocupada por el grafo y la superficie necesaria para una inmersión [BET99]. Cada situación presenta características específicas propias que se traducen en determinadas condiciones sobre el trazado del grafo.

El Teorema de Kuratowski marca, en cierta medida, el inicio de esta disciplina [kur30]. Data de 1930 y caracteriza los grafos que admiten una inmersión en el plano, i.e. una representación plana sin que dos aristas se crucen (excepto en un vértice). Ya en esos años se sabía que los grafos no planos poseían inmersiones en otras superficies. El grafo completo K_5 y el grafo bipartito completo $K_{3,3}$ son dos ejemplos de grafos no planos que admiten inmersiones en el toro, una superficie orientable de género $g = 1$. La Figura 4.1 muestra una inmersión ortogonal de $K_{3,3}$ en el toro (a la izquierda) y la correspondiente representación ortogonal plana estándar (a la derecha), con sus aristas formadas por segmentos de recta horizontales y verticales con un único codo como máximo.

Desde un punto de vista práctico, diversas aplicaciones hicieron un uso temprano de los resultados teóricos obtenidos, pero no es hasta finales de los años 60 cuando empezó a surgir la necesidad de algoritmos

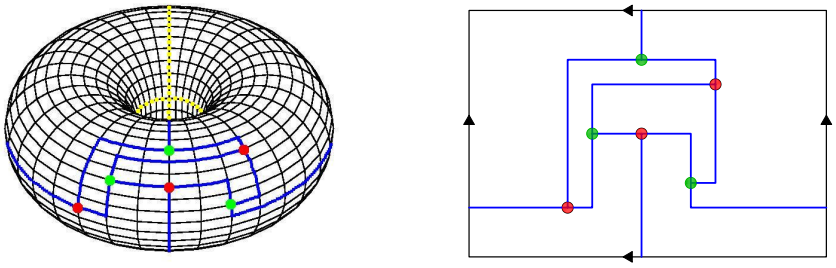


Figura 4.1: Representación gráfica del grafo bipartito completo $K_{3,3}$ sobre el toro, con aristas rectilíneas con un codo y sobre la correspondiente representación ortogonal plana de la superficie.

para la representación de grafos. Esto se debió a una de las aplicaciones principales del trazado de grafos: el diseño de circuitos electrónicos. Es en esta época donde el incremento, cada vez mayor, del número de elementos que componían un circuito comenzaron a hacer el diseño demasiado complicado. En el libro de Lengauer podemos encontrar una recopilación de estos primeros algoritmos para el diseño de redes y circuitos [len90]. Fue también en esta década quando Hopcroft y Tarjan (1974) obtuvieron la aplicación práctica del Teorema de Kuratowski de planaridad, dando una caracterización que condujo al diseño de un algoritmo efectivo [HT74].

La representación de grafos con aristas constituidas por secuencias de segmentos rectilíneos horizontales y verticales ha sido utilizada en diversas aplicaciones, principalmente en los trazados de circuitos, organigramas y diagramas de flujo [BNT86, BTT84]. Este hecho ha atraído la atención de muchos autores y han sido obtenidos numerosos resultados acerca de representaciones ortogonales [bie96, GM98, LMS91, LMS98, RCS86, tam87].

Una aproximación al diseño VLSI de circuitos integrados consiste en considerar el circuito a diseñar como un conjunto de pares de vértices que deben unirse mediante aristas ortogonales, con similares restricciones que en el problema general del trazado ortogonal de grafos, i.e., es conveniente minimizar el número de codos y el número de cruces.

En este sentido se plantea el problema EMPAREJAMIENTO ORTOGONAL SIMPLE PLANO (EOSP), en el cual parejas de puntos en el plano deben unirse mediante trazados ortogonales con un único codo sin que éstos se crucen.

Este problema, EOSP, designado originariamente en inglés por *Single Bend Wiring* (SBW), fue estudiado y resuelto por Raghavan *et al.* [RCS86]. Para ello, estos autores desarrollaron un algoritmo cua-

drático respecto al número de pares de puntos.

Dado el reducido número de posibilidades (sólo dos) de conectar cada par de puntos en el plano, resulta imposible en muchas instancias conectar todos los pares de puntos sin que haya cruces entre aristas. Para estos casos, los mismos autores mostraron que el problema MAX-EOSP (en el original MAX-SBW), i.e., determinar la cardinalidad del mayor subconjunto de pares de puntos que pueden ser conectados sin cruces de aristas, es un problema NP-duro.

Para una instancia del problema que no admita solución en el plano, se procura solucionar el problema trazando las aristas ortogonales en distintas placas paralelas. De nuevo, los mismos autores probaron que esta estrategia nos lleva a un problema NP-completo. Estos resultados, así como los resultados acerca de trazados ortogonales presentados Cohoon y Heck [CH88] y por Yen [yen94], motivaron a Garrido *et al.* [GMM02] y a Portillo [por02] a considerar el problema de emparejamiento ortogonal simple en otras superficies. El trazado de circuitos es

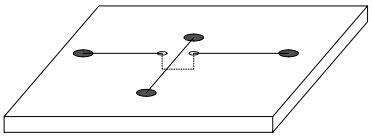


Figura 4.2: Esquema de una asa en un circuito integrado.

un problema de optimización en el cual pueden ser consideradas varias funciones objetivo con intereses relevantes [len90]. El número de *asas* y el número de codos por arista constituyen dos de estas funciones objetivo a considerar. Un asa modela un cambio de cara (*layer*) en el circuito utilizada para evitar cruces entre aristas (Figura 4.2). Al igual que sucede con los codos, los cambios de caras introducen inestabilidad eléctrica perturbando el funcionamiento del circuito. Por otro lado, la utilización de asas aumenta el grado de dificultad en los procesos de fabricación de los circuitos. Considerando los efectos adversos, tanto el número de asas como el número de codos deben minimizarse.

Imponiendo como restricciones al problema de conectar un conjunto de pares de puntos con aristas ortogonales que cada arista tenga como máximo un codo, resulta que cada arista tendrá longitud mínima y todo el trazado ocupará área mínima. En estas condiciones la existencia de asas es inevitable en la mayor parte de los casos. Al considerar el trazado de circuitos en superficies de género g, Garrido, Márquez, Morgana y Portillo tratan el problema de la minimización del número de asas de la siguiente forma: de cada vez que dos aristas se cruzan, esa intersección puede ser evitada introduciendo una asa. Cada vía equivale a un aguje-

ro en la placa donde se traza el circuito, asociado a un cambio de cara en la placa. Entonces, el problema del trazado de circuitos puede ser modelado como un problema de emparejamiento ortogonal en superficies. Cada nueva asa corresponde a aumentar en 1 el género de la superficie. El problema de minimización del género de una superficie en la cual exista un emparejamiento ortogonal simple para un conjunto de pares de puntos dado es un problema NP-duro [GMM02].

También en este ámbito, los mismos autores probaron que dados una superficie y un conjunto de pares de puntos sobre la superficie (de género g), determinar la existencia de un emparejamiento ortogonal simple es un problema NP-completo.

Para superficies particulares la complejidad del problema de emparejamiento ortogonal simple no ha sido establecida hasta ahora. En el presente trabajo, demostraremos que el problema cilíndrico, a semejanza del problema EOSP, es también resoluble en tiempo polinomial. A este efecto, reduciremos las instancias del problemas geométrico a instancias del problema de satisfacibilidad para probar que EOSC-SAT es una subclase de 2PURL. También presentaremos un algoritmo polinomial que permite resolver las instancias del problema EOSC.

Visto el resultado establecido y observando la semejanza entre este problema y el problema de emparejamiento ortogonal simple en el toro, se conjetura que este último también pueda ser resuelto en tiempo polinomial.

4.1. Introdución

Dado un grafo $G = (V, A)$, un *emparejamiento* M en G es un subconjunto de aristas que no son adyacentes dos a dos, i.e. dos aristas de M no inciden en el mismo vértice o, equivalentemente, los vértices del grafo (V, M) tienen valencia máxima uno. Considerando los vértices distribuidos sobre una superficie, un *emparejamiento ortogonal* es una representación gráfica de un emparejamiento de modo que cada arista está formada por una secuencia de segmentos de recta ortogonales entre sí, paralelos a un sistema ortogonal de referencia. Dado un emparejamiento, saber si existe un emparejamiento ortogonal sin que dos aristas se crucen es un interesante problema de decisión. Si cada arista está compuesta a lo máximo por un codo diremos que el problema de emparejamiento ortogonal es *simple*.

Uno de los casos particulares, el problema del EMPAREJAMIENTO ORTOGONAL SIMPLE PLANO ((EOSP) (en inglés *Single Bend Wiring* o

SBW) fue estudiado y resuelto por Raghavan, Cohoon y Sahni [RCS86]. Para resolverlo, cada instancia W se transforma en tiempo polinomial en una fórmula proposicional $\mathcal{F}$ que resulta pertenecer siempre a 2SAT. La instancia tiene solución afirmativa si y sólo si la fórmula proposicional correspondiente es satisfacible. En caso afirmativo, las atribuciones de verdad que satisfacen la fórmula proposicional permiten, también, identificar emparejamientos ortogonales en el conjunto de pares de puntos de W. Dado que una fórmula proposicional 2SAT puede ser resuelta por cualquiera de los algoritmos lineales conocidos (en tiempo $O(|\mathcal{F}|)$), el problema de decisión puede ser resuelto en tiempo polinomial. Dado que la referida reducción se puede efectuar en tiempo cuadrático respecto al número de pares de puntos, en total el problema EOSP (SBW) puede ser resuelto en tiempo $O(n^2)$.

Estudiando el problema de EMPAREJAMIENTO ORTOGONAL EN SUPERFICIES genéricas (EOS), Garrido, Márquez, Morgana y Portillo probaron que se trataba de un problema NP-Completo [GMM02, por02]. Todavía, la complejidad del problema para los problemas de emparejamiento ortogonal en cada una de las superficies particulares, distintas del plano, permanecía como problema abierto.

En el presente capítulo mostraremos que el problema de EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL CILINDRO (EOSC) puede ser resuelto en tiempo polinomial. La demostración de este resultado se hará mediante la reducción del problema a una clase de satisfacibilidad EOSC-SAT y comprobando que ésta es una subclase de la clase de satisfacibilidad polinomial 2PURL.

§ 4.1.1. Reducción de las instancias del problema EOSC a fórmulas proposicionales.— Dados un par de puntos w_i de una instancia W del problema EOSC, en general existen cuatro distintas aristas ortogonales simples posibles que los unen. Cada una de estas aristas puede ser codificado mediante un par de variables lógicas (h_i, m_i), dónde $h_i = 1$ si el segmento incidente en el punto de mayor ordenada del par w_i es horizontal y $m_i = 1$ si el trazado corta al meridiano exterior de la representación ortogonal estándar adoptada, conforme se ilustra en la Figura 4.3.

Fijemos la notación. Sea $w \in W$ un par de puntos de una instancia del problema EOSC, $w = (a, b)$, con coordenadas $a = (a_x, a_y)$ y $b = (b_x, b_y)$, respectivamente, de modo que $a_y > b_y$. Dados dos pares de puntos, w_i y w_j, diremos que $w_i < w_j$ ($i < j$) si el punto de mayor ordenada del par w_i está encima del punto de mayor ordenada del par w_j, i.e. $a_y^i > a_y^j$.

Con el objeto de desarrollar una codificación para las instancias del problema, observemos que, para dos pares de puntos, el conjunto de aristas ortogonales que los unen, sin cruces, no dependen de los valo-

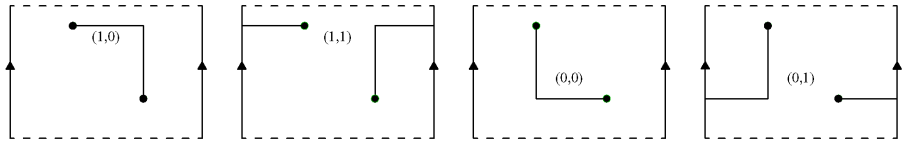


Figura 4.3: Codificación de las cuatro aristas ortogonales que pueden unir un par de puntos en el cilindro.

res de las coordenadas de cada uno de los puntos sino de su posición relativa. A los efectos de esta codificación, consideremos las coordenadas de los puntos de una instancia W del problema EOSC en dos secuencias: de las ordenadas o y-secuencia, $(a_y^1, b_y^1, a_y^2, b_y^2, \cdots, a_y^n, b_y^n)$, con $a_y^i > a_y^{i+1}$ $(i = 1, \cdots, n)$ y $a_y^i > b_y^i$, y de las abscisas o x-secuencia, $(a_x^1, b_x^1, a_x^2, b_x^2, \cdots, a_x^n, b_x^n)$. Tomemos una rejilla ortogonal en la representación ortogonal estándar del cilindro cuyas intersecciones coincidan con cada uno de los puntos de cada uno de los pares $w_i \in W$ (Figura 4.6). Numerando sus líneas verticales con enteros de 1 hasta n, de la izquierda a la derecha y las líneas horizontales de bajo hasta arriba, cada una de las secuencias anteriores puede ser substituida por una secuencia de enteros, $(y_a^1, y_b^1, \cdots, y_a^n, y_b^n)$ y $(x_a^1, x_b^1, \cdots, x_a^n, x_b^n)$, respectivamente.

Supongamos, a titulo de ejemplo, que w_i y w_j son dos pares de puntos para los cuales $a_y^i > a_y^j > b_j^i > b_y^j$ y $b_x^j < a_x^i < a_x^j < b_x^i$, como se ilustra en la Figura 4.4. Considerando la rejilla ortogonal anterior o equivalente, representando la menor de las abscisas (resp. ordenadas) por 1, la segunda por 2, la tercera por 3 y la cuarta por 4 y substituyendo en las respectivas y y x secuencias se obtiene, respectivamente, $(4, 2, 3, 1)$ y $(2, 4, 3, 1)$. Así, este par de secuencias de enteros constituye una representación para todos los pares de puntos cuyas coordenadas satisfagan la ordenación atrás indicada y, por eso, son topológicamente equivalentes.

De este modo, para analizar todas las entradas con dos puntos es suficiente estudiar las configuraciones topológicamente distintas, correspondiente a las 3 y-secuencias (aquellas para las que $y_a^i > y_b^i$, $y_a^j > y_b^j$ y $y_a^i > y_a^j$) y las $4!$ x-secuencias posibles. En total hay $3 \cdot 4! = 72$ secuencias y el mismo número de configuraciones posibles.

Las 3 y-secuencias referidas son $(4, 1, 3, 2), (4, 2, 3, 1)$ y $(4, 3, 2, 1)$. Consideremos ahora las x-secuencias. Para el par $w_i = (a^i, b^i)$ pueden darse dos casos, $x_a^i < x_b^i$ y $x_a^i > x_b^i$. Para cada uno de estos casos, el punto a^j (del par w^j) puede ocupar 3 posiciones correspondientes a $x_a^j < min\{x_a^i, x_b^i\}$, $x_a^j > max\{x_a^i, x_b^i\}$ y x_a^j entre los valores de x_a^i y x_b^i. Finalmente, para cada uno de los 2×3 casos anteriores, el punto b^j podrá

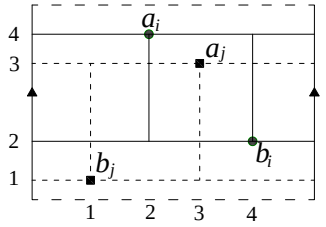


Figura 4.4: Dos pares de puntos de una instancia de EOSC correspondientes a las y y x-secuencias de enteros, (4,2,3,1) y (2,4,3,1), respectivamente.

ocupar 4 posiciones. Por lo tanto, en total hay $4!$ posibles x-secuencias.

Analizando cada una de las antedichas secuencias y codificándolas utilizando las variables booleanas referidas anteriormente, Portillo (2002) identificó doce fórmulas proposicionales del problema de satisfacibilidad proposicional SAT a las cuales se reducen cualquier instancia del problema geométrico, EOSC [por02]. Estas doce fórmulas elementales se presentan en el Cuadro 4.1, numeradas del I al XII, para futuras referencias. El cuadro siguiente, Cuadro 4.2, muestra la lista de las secuencias y las correspondientes fórmulas proposicionales elementales, identificadas por su número. Las entradas con y-secuencia (4,3,2,1) se han omitido pues a todas ellas les corresponde la fórmula elemental I, que es vacía.

Considerando en la Figura 4.5 un par de puntos por cada rectángulo, localizados en los extremos de una de sus diagonales, se obtiene una representación esquemática de las instancias del problema EOSC con dos pares de puntos. Estas instancias quedan tipificadas por las configuraciones de la figura. Las entradas en la fila superior provienen de configuraciones con y-secuencia $(4, 2, 3, 1)$ y las de la fila inferior provienen de configuraciones con y-secuencia $(4, 1, 3, 2)$. Así, a las entradas representadas en (a) les corresponden fórmulas elementales de IX a XII; a las representadas en (b), fórmulas de VI a VIII; a las de (c), III o IV y las tipo (d) son II o V.

Consideremos ahora una entrada del problema EOSC constituida por n pares de puntos ordenados. Para transformarla en una fórmula proposicional es necesario analizar todas las interacciones entre dos pares de puntos, i.e., cada par de puntos debe compararse con todos los demás pares identificando así $\dfrac{n(n-1)}{2}$ fórmulas elementales. La fórmula proposicional que representa la entrada completa corresponde a la conjun-

Cuadro 4.1: Fórmulas proposicionales elementales del problema EOSC ($i < j$).

	$\mathcal{F}_{i,j}$
I	$\{\}$
II	$\{[m_j]\}$
III	$\{[h_i, \overline{m_j}], [\overline{h_i}, m_j]\}$
IV	$\{[h_i, m_j], [\overline{h_i}, \overline{m_j}]\}$
V	$\{[\overline{m_j}]\}$
VI	$\{[h_i, \overline{m_i}], [\overline{h_j}, \overline{m_j}]\}$
VII	$\{[h_i, \overline{m_i}], [\overline{h_j}, m_j]\}$
VIII	$\{[h_i, m_i], [\overline{h_j}, \overline{m_j}]\}$
IX	$\{[h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j], [h_i, \overline{m_j}, \overline{h_j}], [\overline{h_i}, m_j, \overline{h_j}]\}$
X	$\{[h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j], [h_i, m_j, \overline{h_j}], [\overline{h_i}, \overline{m_j}, \overline{h_j}]\}$
XI	$\{[h_i, m_i, h_j], [h_i, \overline{m_i}, \overline{h_j}], [h_i, \overline{m_j}, \overline{h_j}], [\overline{h_i}, m_j, \overline{h_j}]\}$
XII	$\{[h_i, m_i, h_j], [h_i, \overline{m_i}, \overline{h_j}], [h_i, m_j, \overline{h_j}], [\overline{h_i}, \overline{m_j}, \overline{h_j}]\}$

ción de todas las fórmulas elementales. Por lo tanto la reducción de cada una de las instancias de EOSC a instancias de SAT puede ser efectuada en tiempo cuadrático, $O(n^2)$. El resultado sigue siendo válido aunque la entrada no esté ordenada pues, en una operación de preprocesamiento, los puntos de W pueden ser ordenados en tiempo $O(nlog(n))$ sin afectar a la complejidad de la reducción. Denotaremos EOSC-SAT a la clase de satisfacibilidad formada por las fórmulas proposicionales que correspondan a alguna entrada del problema de emparejamiento ortogonal. Obviamente, EOSC-SAT es una subclase propia de 3SAT, siendo esta última una clase NP-completa.

La Figura 4.6 muestra una instancia de EOSC formada por 4 pares de puntos. Analizando el par w_1 con los demás, w_2, w_3 y w_4, se comprueba que a la primera interacción le corresponde la y-secuencia $(4, 2, 3, 1)$ y la x-secuencia $(1, 3, 4, 2)$, por lo tanto una fórmula elemental IX, a la segunda la misma y-secuencia y la x-secuencia $(2, 4, 3, 1)$, por lo tanto una fórmula XII y a la tercera una fórmula vacía (fórmula I, y-secuencia $(4, 3, 2, 1)$), respectivamente. Repitiendo el procedimiento, analizando ahora el par w_2 con w_3 y w_4, a w_2, w_3 le corresponde la fórmula XI y a w_2, w_4, la fórmula I. Finalmente a w_3, w_4 le corresponde una fórmula elemental IV. La conjunción de las fórmulas elementales anteriores (en el orden indicado) corresponde a la fórmula proposicio-

Cuadro 4.2: Secuencias para dos pares de puntos de una instancia del problema EOSC y sus correspondientes fórmulas elementales.

y-seq.	x-seq.	Fórmula	y-seq.	x-seq.	Fórmula
(4,2,3,1)	(1,2,3,4)	VI	(4,1,3,2)	(1,2,3,4)	V
	(1,2,4,3)	VI		(1,2,4,3)	V
	(1,3,2,4)	XI		(1,3,2,4)	III
	(1,3,4,2)	IX		(1,3,4,2)	III
	(1,4,2,3)	VIII		(1,4,2,3)	V
	(1,4,3,2)	VIII		(1,4,3,2)	V
	(2,1,3,4)	VI		(2,1,3,4)	V
	(2,1,4,3)	VI		(2,1,4,3)	V
	(2,3,1,4)	VII		(2,3,1,4)	II
	(2,3,4,1)	VII		(2,3,4,1)	II
	(2,4,1,3)	X		(2,4,1,3)	IV
	(2,4,3,1)	XII		(2,4,3,1)	IV
	(3,1,2,4)	XII		(3,1,2,4)	IV
	(3,1,4,2)	X		(3,1,4,2)	IV
	(3,2,1,4)	VII		(3,2,1,4)	II
	(3,2,4,1)	VII		(3,2,4,1)	II
	(3,4,1,2)	VI		(3,4,1,2)	V
	(3,4,2,1)	VI		(3,4,2,1)	V
	(4,1,2,3)	VIII		(4,1,2,3)	V
	(4,1,3,2)	VIII		(4,1,3,2)	V
	(4,2,1,3)	IX		(4,2,1,3)	III
	(4,2,3,1)	XI		(4,2,3,1)	III
	(4,3,1,2)	VI		(4,3,1,2)	V
	(4,3,2,1)	VI		(4,3,2,1)	V

nal (4.1), equivalente a la instancia del problema EOSC y en la que los literales asociados a cada una de las variables (h_i, m_i) están representados por los enteros $2i$ y $2i-1$ respectivamente.

$$\begin{aligned}\mathcal{F} = \quad & \{[1,2,-4],[-1,2,4],[2,-3,-4],[-2,3,-4],\\ & [1,2,6],[-1,2,-6],[2,5,-6],[-2,-5,-6],\\ & [3,4,6],[-3,4,-6],[4,-5,-6],[-4,5,-6],\\ & [6,7],[-6,-7]\}\end{aligned} \tag{4.1}$$

En todos los ejemplos que siguen consideraremos las fórmulas elementales ordenadas de esta manera y los literales representados por enteros siguiendo la notación que acabamos de referir.

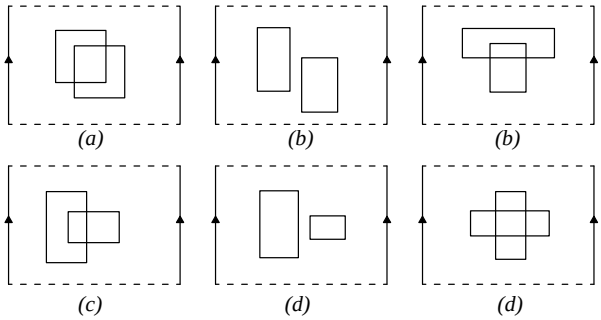


Figura 4.5: Representación tipificada de instancias de pares de puntos y respectivas fórmulas elementales: (*a*) IX a XII (3SAT), (*b*) VI a VIII, (*c*) III o IV y (*d*) II o V. (Los puntos de cada par están en esquinas opuestas de cada rectángulo.)

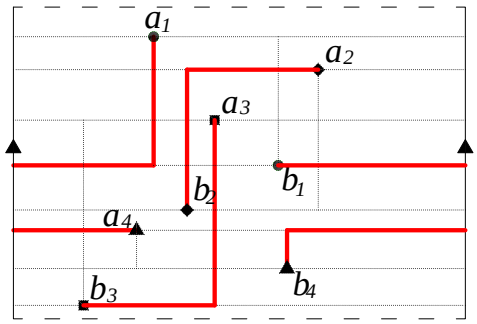


Figura 4.6: Ejemplo de una instancia de EOSC con cuatro pares de puntos con respuesta afirmativa y un emparejamiento ortogonal simple que la resuelve.

Detectando y eliminado los literales p-eliminables secuencialmente mediante el algoritmo ELIMINALITERALES (v. Algoritmo 3) se obtiene la fórmula proposicional $\mathcal{F}^E$ lógicamente equivalente a $\mathcal{F}$.

$$\begin{aligned}\mathcal{F}^E = \quad & \{[1,2],[2,4],[2,-3],[-2,3,-4],\\ & [1,2],[2,-6],[2,-6],[-5,-6],\\ & [3,4,6],[-3,-6],[-5,-6],[-4,-6],\\ & [6,7],[-6,-7]\}\end{aligned}$$

Se comprueba fácilmente que la atribución de verdad $\tau = \{1,-2,-3,4,-5,-6,7,8\}$ es un modelo para $\mathcal{F}^E$ y que por lo tanto esta fórmula proposicional es satisfacible.

Como $\mathcal{F}$ y $\mathcal{F}^E$ son lógicamente equivalentes, la fórmula original es

satisfacible y τ uno de sus modelos.

Además de obtener la resolución del problema EOSC, en caso de respuesta afirmativa las atribuciones de verdad satisfaciendo la fórmula proposicional permiten identificar los emparejamientos ortogonales soluciones del problema EOSC. Ilustrando este hecho, la Figura 4.6 muestra el emparejamiento correspondiente a la atribución de verdad τ. En resumen, la codificación establecida define una correspondencia unívoca entre cada una de las 4 aristas ortogonales simples por cada par de puntos y las atribuciones de verdad del par de variables lógicas correspondientes.

4.2. EOSC-SAT *no es subclase de las clases de satisfacibilidad polinomiales* bien conocidas

Recordemos que, como se ha referido anteriormente en la Sección 1.7, las clases polinomiales SLUR y LinAut, que son incomparables entre sí, contienen entre ambas a todas las demás clases de satisfacibilidad proposicional polinomiales bien conocidas (v. Figura 1.17). Entonces, para comprobar que la clase de satisfacibilidad EOSC-SAT no es subclase de ninguna de las clases polinomiales bien conocidas es suficiente probar que existe una fórmula proposicional de una instancia del problema EOSC que no pertenezca ni a SLUR ni a LinAut.

Para ello consideraremos la instancia del problema EOSC representada en la Figura 4.7 y la correspondiente fórmula proposicional (4.2).

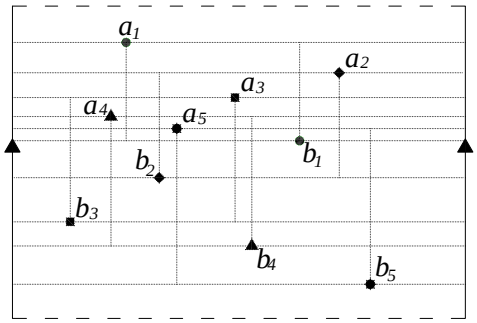


Figura 4.7: Entrada del problema EOSC cuya fórmula proposicional no es instancia de SLUR ni de LinAut (v. ecuación (4.2)).

$$\begin{aligned}\mathcal{F} = \quad & \{[1,2,-4],[-1,2,4],[2,-3,-4],[-2,3,-4],\\ & [1,2,6],[-1,2,-6],[2,5,-6],[-2,-5,-6],\\ & [1,2,-8],[-1,2,8],[2,7,-8],[-2,-7,-8],\\ & [1,2,10],[-1,2,-10],[2,-9,-10],[-2,9,-10],\\ & [3,4,6],[-3,4,-6],[4,-5,-6],[-4,5,-6],\\ & [3,4,-8],[-3,4,8],[4,-7,-8],[-4,7,-8],\\ & [3,4,10],[-3,4,-10],[4,9,-10],[-4,-9,-10],\\ & [5,6,8],[-5,6,-8],[6,7,-8],[-6,-7,-8],\\ & [5,6,10],[-5,6,-10],[6,9,-10],[-6,-9,-10],\\ & [7,8,10],[-7,8,-10],[8,-9,-10],[-8,9,-10]\}\end{aligned} \tag{4.2}$$

Si se ejecuta el algoritmo SLUR (v. Algoritmo 2) con entrada la fórmula proposicional $\mathcal{F}$ (v. ecuación 4.2), comenzando por la variable 3 seguida de la variable 5 y en la opción arbitraria, escogiendo los literales negativos, este algoritmo termina con el mensaje *desisto*. Por lo tanto, $\mathcal{F}$ no es una instancia de la clase SLUR.

Para comprobar que $\mathcal{F}$ tampoco es una instancia de LinAut es suficiente comprobar que, para la matriz cláusula–variable A de $\mathcal{F}$ (4.3), la condición $A\mathbf{x} \geq \mathbf{0}$ tiene como única solución $\mathbf{x} = \mathbf{0}$.

Comenzamos sumando ordenadamente las inecuaciones correspondientes a las líneas 3 y 4 de la matriz, se obtiene la condición $-2x_4 \geq 0$. De un modo similar, a partir de los pares de líneas 7 y 8, y 11 y 12, se obtienen las condiciones $-2x_6 \geq 0$ y $-2x_8 \geq 0$, respectivamente. Combinando ahora las líneas 13 y 14, 25 y 26, 29 y 30 y 37 y 38, se tienen las condiciones $2x_2 \geq 0$, $2x_4 \geq 0$, $2x_6 \geq 0$ y $2x_8 \geq 0$, respectivamente. Conjugando estas desigualdades con las anteriores se tiene que $x_4 = x_6 = x_8 = 0$. Sumando ordenadamente las inecuaciones asociadas a las líneas 17 y 22 de la matriz A, tenemos ahora la desigualdad $2x_3 + 2x_4 + x_6 - x_8 \geq 0$, de dónde $x_3 \geq 0$. Con la misma construcción, de las líneas 18 y 23 concluimos que $-x_3 \geq 0$, por lo que $x_3 = 0$. De las inecuaciones 19 y 20, $x_4 - x_5 - x_6 \geq 0$ y $-x_4 + x_5 - x_6 \geq 0$ se deduce $x_5 = 0$. Análogamente, a partir de la líneas 31 y 32 se concluye que $x_7 = 0$. Por lo tanto, hasta este momento, se ha deducido que todas las variables $x_3, \cdots, x_8$ son nulas. Continuando con el mismo procedimiento, a partir de los pares de líneas 3 y 4, 1 y 2, 37 y 38, y 39 y 40 resulta que $x_2 = 0$, $x_1 = 0$, $x_{10} = 0$ y $x_9 = 0$. Se concluye, por tanto, que la condición $A\mathbf{x} \geq \mathbf{0}$ tiene como única solución el vector nulo $\mathbf{x} = \mathbf{0}$, por lo que la fórmula proposicional $\mathcal{F}$ no pertenece a la clase de satisfacibilidad polinomial LinAut.

A partir de los resultados anteriores, se deduce que la fórmula proposicional $\mathcal{F}$ no es instancia ni de SLUR ni de LinAut, por lo que EOSC-SAT

$$A = \begin{pmatrix}
1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1
\end{pmatrix} \tag{4.3}$$

no es subclase de ninguna de las clases de satisfacibilidad polinomiales bien conocidas.

Para terminar nótese que, en este caso, el algoritmo ELIMINALITERALES retorna una fórmula proposicional $\mathcal{F}^E$ constituida exclusivamente por cláusulas unitarias,

$$\mathcal{F}^E = \quad \{[2], [3], [4], [5], [-6], [7], [-8], [-10]\}. \tag{4.4}$$

cualquier atribución de verdad conteniendo la atribución $\tau = \{2, 3, 4, 5, -6, 7, -8, 10\}$ será un modelo para $\mathcal{F}^E$. Y, por consiguiente, también para $\mathcal{F}$ pues estas dos fórmulas son lógicamente equivalentes.

Además, como cada cláusula de $\mathcal{F}$ contiene un literal p-eliminable (en $\mathcal{F}$) de acuerdo al Lema 2.5 y a la Definición 2.4, $\mathcal{F}$ es una instancia de Φ_2 y, por lo tanto, de PURL.

4.3. Grafo de interacciones

La definición de grafo de interacciones, asociado a una instancia del problema de emparejamiento EOSC, es idéntica a la presentada para el problema de etiquetado UC-4P. A este efecto, considerando una entrada W del problema de emparejamiento ortogonal simple en el cilindro (EOSC), se llama *grafo de interacciones* al grafo finito, $G = (V, A)$, dónde a cada vértice $v_i \in V$ le corresponde el par de puntos $w_i \in W$. La arista $\{v_i, v_j\} \in A$ si a los pares de puntos w_i y w_j les corresponde una de las fórmulas proposicionales elementales II a XII del Cuadro 4.1. Lo que sucede si y sólo si $y_a^i > y_a^j > y_b^i$. Los vértices de V se consideran ordenados mediante la ordenación inducida por los elementos del conjunto W, i.e., $v_i < v_j$ si y sólo si $w_i < w_j$.

$\mathcal{F}$ admite una partición en variables disjuntas si existen subfórmulas $\mathcal{F}^i$, $i = 1, \cdots, k$ de modo que $var(\mathcal{F}^i)$ y $var(\mathcal{F}^j)$ son conjuntos disjuntos y $\mathcal{F} = \mathcal{F}^1 \cup \mathcal{F}^2 \cup \cdots \cup \mathcal{F}^k$. En el caso de que el grafo de interacciones no fuese conexo, la fórmula proposicional $\mathcal{F}$ admite una partición en variables disjuntas con una subfórmula por cada componente conexa del grafo. En estas condiciones la fórmula proposicional $\mathcal{F}$ es satisfacible si y sólo si cada una de sus componentes son satisfacibles y, en este caso, la resolución de la fórmula puede efectuarse resolviendo cada una de las referidas subfórmulas independientemente de las demás. Por este hecho, en el estudio que sigue, consideraremos solamente entradas del problema EOSC cuyo grafo de interacciones sea conexo.

Consideremos el conjunto de vértices del grafo de interacciones (conexo) $V = \{v_1, \cdots, v_n\}$, ordenados por orden creciente de sus índices y designemos por $N(v_n) = \{v_i \in V : v_i < v_n \text{ y } \{v_i, v_n\} \in A\}$ al conjunto de vértices adyacentes al mayor vértice de V, v_n. Como probaremos a continuación, el subgrafo inducido $G(N(v_n) \cup \{v_n\})$ es un grafo completo.

Suponiendo que existen dos vértices adyacentes al mayor vértice de V, v_n, las correspondientes ordenadas verifican las siguientes desigualdades: $y_a^i > y_a^n > y_b^i$ y $y_a^j > y_a^n > y_b^j$. Considerando, sin pérdida de generalidad, que $v_i < v_j$, $y_a^i > y_a^j$ y, combinando con las dos desigualdades anteriores, resulta que $y_a^i > y_a^j > y_a^n > y_b^j$. De dónde se concluye que los vértices v_i y v_j son también adyacentes entre sí pues $y_a^i > y_a^j > y_b^i$. Lo que prueba que el subgrafo inducido $G(N(v_n) \cup \{v_n\})$ es un grafo completo y, por lo tanto, v_n es un vértice simplicial (v. Sección 1.1.2).

Consideremos el subconjunto de vértices $V_1 = V \backslash \{v_n\}$, el subgrafo inducido $G_1 = G(V_1)$ y el mayor vértice de V_1, v_{n-1}. Por el mismo razonamiento que anteriormente se prueba que v_{n-1} es, también, un vértice simplicial. Repitiendo de nuevo el mismo argumento con los sucesivos subconjuntos se concluir que $v_n, v_{n-1}, \cdots, v_1$, es una secuencia de eli-

minación perfecta, por lo que G es un grafo cordal [FG65]. Este hecho permite enunciar el siguiente resultado:

Lema 4.1. *El grafo de interacciones de una instancia del problema* EOSC *es un grafo cordal.*

En general, para cada vértice $v_k \in V$, consideraremos el conjunto de vértices adyacentes subdividido en dos conjuntos: $N(v_k) = \{v_i \in V : \{v_i, v_k\} \in A \text{ y } v_i < v_k\}$ y $N^*(v_k) = \{v_i \in V : \{v_i, v_k\} \in A \text{ y } v_i > v_k\}$.

Definición 4.1. Sea W una instancia del problema EOSC. Un par de puntos $w_i \in W$ se llama *condicionado* si uno de los caminos con salida horizontal de w_i corta los dos segmentos verticales de otro par $w_j \in W$ y uno de los caminos con salida vertical corta los dos segmentos verticales de otro par $w_k \in W$, pudendo darse $j = k$.

Como veremos más adelante, la existencia de un par de puntos condicionados implica la simplificación de algunas de las cláusulas 3SAT asociadas a ese par de puntos, puesto que algunos de sus literales son p-eliminables.

Los vértices del grafo de interacciones correspondientes a pares condicionados se denominan *vértices condicionados*.

En la instancia del problema EOSC presentada en la Figura 4.8, los pares w_2 y w_4 son ejemplos de pares condicionados. En el primer caso, w_2, uno de los dos caminos con salida horizontal, $(1, 0)$, corta a los dos segmentos verticales de los posibles caminos que unen a los puntos del par w_1 y el camino con salida vertical $(0, 1)$ corta a los dos segmentos verticales del par w_3. En el segundo de los pares referidos, los dos caminos del par w_4 que cortan al meridiano exterior, uno de ellos con salida vertical y el otro con salida horizontal, intersecan a los dos segmentos verticales de los posibles caminos del par w_3.

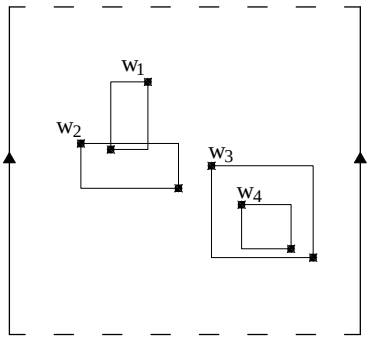


Figura 4.8: w_2 y w_4 son *pares de puntos condicionados*; pero w_1 y w_3, no.

En un grafo de interacciones, $v_i \in V$ es un vértice condicionado si se verifica alguna de las siguientes condiciones:

- existe un vértice $v_a \in N(v_i)$ cuya fórmula proposicional $\mathcal{F}_{a,i}$ es una fórmula II o V
- existen vértices, $v_a \in N(v_i)$ y $v_b \in N^*(v_i)$ de modo que cada una de las fórmulas elementales $\mathcal{F}_{a,i}$ y $\mathcal{F}_{i,b}$ le corresponde una fórmula del tipo VI a VIII.

Para cada una de las fórmulas elementales VI a XII consideraremos la partición $\mathcal{F}_{ij} = \mathcal{F}_{ij}^T \cup \mathcal{F}_{ij}^B$ dónde cada una de las cláusulas de $\mathcal{F}_{ij}^T$ contienen uno de los literales sobre la variable m_i y cada una de las cláusulas de $\mathcal{F}_{ij}^B$ uno de los literales sobre la variable m_j.

La existencia de vértices condicionados en el grafo de interacciones está asociada a la existencia de cláusulas con literales p-eliminables lo que permite la simplificación de las fórmulas proposicionales.

A continuación se presentan los lemas 4.2 a 4.9, los cuales serán herramientas utilizadas para probar que EOSC-SAT es subclase de 2PURL.

Lema 4.2. *Sea $\mathcal{F}_{i,j}$ $(i < j)$ una fórmula proposicional elemental* 3SAT *(fórmula del tipo* IX *a* XII*) y $\mathcal{F}_{i,k}$ $(i < k)$ una fórmula* VI *a* VIII*. Entonces cada una de las cláusulas de la subfórmula $\mathcal{F}_{i,j}^T$ contiene un literal p-eliminable.*

Demostración. En las condiciones indicadas, para cada una de las fórmulas elementales IX a XII, $\mathcal{F}_{i,j}^T = \{[h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j]\}$ o $\mathcal{F}_{i,j}^T = \{[h_i, m_i, h_j], [h_i, \overline{m_i}, \overline{h_j}]\}$ y, para cada una de las fórmulas de los tipos VI a VIII, $[h_i, m_i] \in \mathcal{F}_{i,k}$ o $[h_i, \overline{m_i}] \in \mathcal{F}_{i,k}$.

Considerando $\mathcal{F}_{i,j}^T = \{[h_i, m_i, \overline{h_j}], [h_i, \overline{m_i}, h_j]\}$ y $[h_i, m_i] \in \mathcal{F}_{i,k}$, el literal $\overline{h_j} \in [h_i, m_i, \overline{h_j}]$ y el literal $\overline{m_i} \in [h_i, \overline{m_i}, h_j]$ son p-eliminables pues la atribución de verdad $\tau = \{\overline{h_j}, \overline{h_i}, \overline{m_i}\}$ falsifica la fórmula elemental $\mathcal{F}_{i,k}$ y por consiguiente la fórmula $\mathcal{F}$.

Utilizando el mismo método para cada una de las otras tres posibilidades se deduce fácilmente la afirmación. □

Lema 4.3. *Sea $\mathcal{F}_{i,k}$ $(i < k)$ una fórmula elemental* 3SAT *y $\mathcal{F}_{j,k}$ $(j < k)$ una fórmula* VI *a* VIII*. Entonces cada una de las cláusulas de la subfórmula $\mathcal{F}_{i,k}^B$ contiene un literal p-eliminable.*

Demostración. La verificación del resultado es una simple comprobación considerando cada uno de los 4 casos posibles atendiendo a que $\mathcal{F}_{i,k}^B = \{[h_i, \overline{m_k}, \overline{h_k}], [\overline{h_i}, m_k, \overline{h_k}]\}$ o $\mathcal{F}_{i,k}^B = \{[h_i, m_k, \overline{h_k}], [\overline{h_i}, \overline{m_k}, \overline{h_k}]\}$ y a que o bien $[\overline{h_k}, m_k] \in \mathcal{F}_{j,k}$, o bien $[\overline{h_k}, \overline{m_k}] \in \mathcal{F}_{j,k}$. □

Lema 4.4. *Consideremos $\mathcal{F}$ una fórmula proposicional conteniendo la fórmula elemental* 3SAT *$\mathcal{F}_{j,k}$ $(j < k)$.*

- *Si $\mathcal{F}_{i,j}$ ($i < j$) es una fórmula elemental* II *o* V *entonces cada cláusula de $\mathcal{F}^T_{j,k}$ contiene un literal p-eliminable;*
- *Si $\mathcal{F}_{i,k}$ ($i < k$) es una fórmula elemental* II *o* V *entonces cada cláusula de $\mathcal{F}^B_{j,k}$ contiene un literal p-eliminable.*

Demostración. La demostración es, de nuevo, una simple comprobación del hecho de que en las condiciones de la hipótesis se verifica que $\mathcal{F}^T_{j,k} = \{[h_j, m_j, \overline{h_k}], [h_j, \overline{m_j}, h_k]\}$ o $\mathcal{F}^T_{j,k} = \{[h_j, m_j, h_k], [h_j, \overline{m_j}, \overline{h_k}]\}$ y también que $\mathcal{F}^B_{j,k} = \{[h_j, \overline{m_k}, \overline{h_k}], [\overline{h_j}, m_k, \overline{h_k}]\}$ o $\mathcal{F}^B_{j,k} = \{[h_j, m_k, \overline{h_k}], [\overline{h_i}, \overline{m_k}, \overline{h_k}]\}$ y que, para una fórmula del tipo II a V, se verifica que $\mathcal{F}_{i,j} = \{[m_j]\}$ o $\mathcal{F}_{i,j} = \{[\overline{m_j}]\}$ y que $\mathcal{F}_{i,k} = \{[m_k]\}$ o $\mathcal{F}_{i,k} = \{[\overline{m_k}]\}$. □

Corolario 4.5. *Sea $G = (V, A)$ el grafo de interacciones de una instancia del problema* EOSC *y $v_i, v_j \in V$ vértices condicionados. Si $\mathcal{F}_{i,j}$ es una fórmula elemental 3*SAT *(de los tipos* IX *a* XII*) entonces cada una de sus cláusulas contiene al menos un literal p-eliminable.*

Demostración. Sin pérdida de generalidad, supongamos que $v_i < v_j$ y consideremos la partición de la fórmula elemental (3SAT) $\mathcal{F}_{i,j} = \mathcal{F}^T_{i,j} \cup \mathcal{F}^B_{i,j}$. Como v_i es un vértice condicionado, existe un vértice adyacente $v_a \in N(v_i)$ de modo que $\mathcal{F}_{a,i}$ es una fórmula elemental II o V o bien existe un vértice $v_b \in N^*(v_i)$ de modo que $\mathcal{F}_{i,b}$ es una fórmula elemental de los tipos VI a VIII. Entonces, de acuerdo al Lema 4.4 o al Lema 4.2, respectivamente, cada cláusula de $\mathcal{F}^T_{i,j}$ contiene un literal p-eliminable.

Por otro lado, dado que v_j es también un vértice condicionado, existe un vértice $v_c \in N(v_j)$ de modo que $\mathcal{F}_{c,j}$ es una fórmula elemental II, V, VI, VII o VIII. En cualquiera de los casos, cada cláusula de $\mathcal{F}^B_{i,j}$ contiene un literal p-eliminable de acuerdo a los lemas 4.4 o 4.3. □

Definición 4.2. Dada una instancia del problema EOSC y el grafo de interacciones $G = (V, A)$, una arista de G, $\{v_i, v_j\} \in A$, se llama una *arista de propagación* si $\mathcal{F}_{i,j}$ es una fórmula elemental III, IV o 3SAT (de IX a XII).

Durante el procesamiento del algoritmo ELIMINALITERALES, la fórmula proposicional y cada una de las lógicamente equivalentes que se obtienen excluyendo sucesivos literales p-eliminables es resuelta con varias atribuciones de verdad a través del algoritmo de resolución unitaria PROPUNIT. Las aristas de propagación y, en particular, las aristas asociadas a las fórmulas elementales 3SAT presentan algunas propiedades que se sintetizan en los cuadros siguientes.

El Cuadro 4.3 muestra la resolución simple, sin propagación unitaria, de cada una de las fórmulas elementales IX a XII con cada una de las atribuciones de verdad $\{\overline{h_i}, m_i\}$, $\{\overline{h_i}, \overline{m_i}\}$, $\{h_j, m_j\}$ y $\{h_j, \overline{m_j}\}$. En cada

una de ellas se obtiene una fórmula conteniendo una cláusula unitaria con un literal sobre la variable h_j o sobre la variable h_i, respectivamente.

Cuadro 4.3: Resolución (simple) de cada una de las fórmulas elementales 3SAT con las atribuciones $\{\overline{h_i}, m_i^*\}$ y $\{h_j, m_j^*\}$ ($i < j$).

Atribución	Fórmula elemental resuelta			
	IX	X	XI	XII
$\{\overline{h_i}, m_i\}$	$\{[h_j], [\overline{m_j}]\}$	$\{[h_j], [m_j]\}$	$\{[\overline{h_j}]\}$	$\{[\overline{h_j}]\}$
$\{\overline{h_i}, \overline{m_i}\}$	$\{[\overline{h_j}]\}$	$\{[\overline{h_j}]\}$	$\{[h_j], [\overline{m_j}]\}$	$\{[h_j], [m_j]\}$
$\{h_j, m_j\}$	$\{[h_i]\}$	$\{[\overline{h_i}], [m_i]\}$	$\{[h_i]\}$	$\{[\overline{h_i}], [\overline{m_i}]\}$
$\{h_j, \overline{m_j}\}$	$\{[\overline{h_i}], [m_i]\}$	$\{[h_i]\}$	$\{[\overline{h_i}], [\overline{m_i}]\}$	$\{[h_i]\}$

El Cuadro 4.4 muestra la resolución simple de cada una de las fórmulas elementales 3SAT con las posibles atribuciones de verdad sobre las variables h_i y h_j. A destacar el hecho de que la resolución de cada una de las fórmulas elementales IX a XII con una atribución de verdad con el literal afirmado h_i falso (conteniendo el literal negado $\overline{h_i}$) siempre devuelve una fórmula con una cláusula unitaria conteniendo uno de los literales sobre la variable m_i y si la atribución de verdad para el literal afirmado h_j es verdadera, la fórmula resuelta contiene una cláusula unitaria con un literal sobre la variable m_j.

Cuadro 4.4: Resolución de las fórmulas elementales 3SAT con cada una de las atribuciones sobre las variables h_i, h_j ($i < j$).

Atribución	Fórmula elemental			
	IX	X	XI	XII
$\{\overline{h_i}, h_j\}$	$\{[m_i], [\overline{m_j}]\}$	$\{[m_i], [m_j]\}$	$\{[\overline{m_i}], [\overline{m_j}]\}$	$\{[\overline{m_i}], [m_j]\}$
$\{\overline{h_i}, \overline{h_j}\}$	$\{[\overline{m_i}]\}$	$\{[\overline{m_i}]\}$	$\{[m_i]\}$	$\{[m_i]\}$
$\{h_i, h_j\}$	$\{[m_j]\}$	$\{[\overline{m_j}]\}$	$\{[m_j]\}$	$\{[\overline{m_j}]\}$
$\{h_i, \overline{h_j}\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$

Antes de proseguir con el principal objetivo de este capítulo, demostrar que la clase de satisfacibilidad EOSC-SAT se puede resolver en tiempo polinomial, consideremos dos resultados preliminares.

A este efecto considérese una fórmula proposicional sin la cláusula nula y sin literales p-eliminables, $\mathcal{F}^E$, lógicamente equivalente a una instancia $\mathcal{F}$ del problema EOSC-SAT y sea C_0 una cláusula con 3 litera-

les de una fórmula elemental $\mathcal{F}^E_{s,t}$ ($s < t$). Entonces $C_0 = [\overline{h_t}, m^*_t, h^*_s]$ o $C_0 = [h_s, m^*_s, h^*_t]$ dependiendo de que $C_0 \in (\mathcal{F}^E_{s,t})^B$ o $C_0 \in (\mathcal{F}^E_{s,t})^T$, respectivamente. Recordemos que m^*_i (resp. h^*_i) designa uno de los literales m_i o $\overline{m_i}$ (resp. h_i o $\overline{h_i}$), sobre la variable m_i (resp. h_i) para cada $i = 1, \cdots, n$. De acuerdo a cada uno de los casos, para la atribución de verdad $\tau^0 = \{h_t, m^*_t\}$ o $\tau^0 = \{\overline{h_s}, m^*_s\}$ respectivamente, considérese la correspondiente propagación unitaria

$$(\mathcal{F}^1, \tau^1) = PropUnit(\mathcal{F}^E \cup \mathcal{U}(\tau^0)). \tag{4.5}$$

La fórmula proposicional obtenida $\mathcal{F}^1$ no contiene la cláusula nula, pues $\mathcal{F}^E$ no contiene literales p-eliminables.

Para la fórmula obtenida y para un vértice $v_\pi \in V$ consideremos la partición,

$$\mathcal{F}^1 = \mathcal{F}^1_{V^B_\pi} \cup \mathcal{F}^1_{V^T_\pi} \cup \mathcal{F}^1_{V^T_\pi, V^B_\pi} \tag{4.6}$$

dónde $V^T_\pi = \{v_i \in V : v_i \leq v_\pi\}$, $V^B_\pi = \{v_i \in V : v_i > v_\pi\}$, $\mathcal{F}^1_{V^B_\pi}$ y $\mathcal{F}^1_{V^T_\pi}$ son fórmulas proposicionales formadas con las cláusulas de $\mathcal{F}^1$ cuyas variables están asociadas a vértices de los conjuntos V^B_π y V^T_π, respectivamente; y $\mathcal{F}^1_{V^T_\pi, V^B_\pi}$ está formada por la cláusulas de $\mathcal{F}^1$ que contienen simultáneamente un literal sobre una variable de un vértice del conjunto V^T_π y otra sobre una variable de un vértice del conjunto V^B_π. En estas condiciones las fórmulas proposicionales $\mathcal{F}^1_{V^B_\pi}$ y $\mathcal{F}^1_{V^T_\pi}$ son variable-disjuntas. i.e., no tienen variables comunes.

Lema 4.6. *Sea $C_0 \in (\mathcal{F}^E_{s,t})^B$ ($s < t$) una cláusula con 3 literales y una partición (4.6) en la que el vértice v_π es v_t. Entonces para cada arista $\{v_i, v_j\} \in A$ con $v_i \in V^T_t$ y $v_j \in V^B_t$, la variable h_i es atribuida por la atribución τ^1 en (4.5).*

Demostración. Dado que $v_i < v_t < v_j$, entonces $y^i_a > y^t_a > y^j_a > y^i_b$. Por lo tanto, el vértice $v_i \in N(v_t)$ y $\{v_i, v_t\}$ es una arista de propagación, i.e. $\mathcal{F}_{i,t}$ es una fórmula elemental III, IV o 3SAT (IX a XII), pues en caso contrario la cláusula $C_0 \in (\mathcal{F}^E_{s,t})^B$ no podría contener 3 literales, según los lemas 4.3 y 4.4.

Por tanto, dado que la variable h_i es atribuida por propagación unitaria de la atribución $\tau^0 = \{h_t, m^*_t\}$ en $\mathcal{F}$ también lo es en $\mathcal{F}^E$ (v. Cuadro 4.3). Por lo que $h_i \in \tau^1$ o $\overline{h_i} \in \tau^1$. □

Lema 4.7. *En las condiciones anteriores, si $C_0 \in (\mathcal{F}^E_{s,t})^B$ ($s < t$ y $|C_0| = 3$) y en la partición (4.6) consideramos $v_\pi = v_t$, entonces $\mathcal{F}^1_{V^T_t, V^B_t}$ es una fórmula vacía o, como mucho, contiene dos cláusulas con dos literales cada de una misma fórmula elemental.*

Demostración. Supongamos que $\mathcal{F}^1_{V_t^T,V_t^B} \neq \emptyset$. Entonces existe una cláusula $C_1 \in \mathcal{F}^1_{e,f}$, con $v_e \in V_t^T$ y $v_f \in V_t^B$ tal que C_1 contiene un literal asociado al vértice v_e y otro asociado al vértice v_f. Entonces, $\mathcal{F}_{e,f}$ es una fórmula elemental III, IV o 3SAT (IX a XII). Cómo $y_a^e > y_a^t > y_a^f > y_b^e$, $v_e \in N(v_t)$ y, de acuerdo al Lema 4.6, la variable h_e está atribuida por la atribución τ^1.

Pero como $\mathcal{F}^1_{V_\pi^T,V_\pi^B} \neq \emptyset$, $\mathcal{F}_{e,f}$ sólo podrá ser una fórmula elemental 3SAT (en caso contrario, $\mathcal{F}^1_{e,f} = \emptyset$) y el literal negado $\overline{h_e} \in \tau^1$ (pues si fuera $h_e \in \tau^1$, $\mathcal{F}^1_{V_\pi^T,V_\pi^B} = \emptyset$). Dado que $\mathcal{F}_{e,f} = \{[h_e, m_e, h_f^*],\ [h_e, \overline{m_e}, \overline{h_f^*}],\ [\overline{h_f}, m_f, h_e^*],\ [\overline{h_f}, \overline{m_f}, \overline{h_e^*}]\}$, como $\overline{h_e} \in \tau^1$, en el caso de que la variable m_f no esté atribuida se tendrá que $\mathcal{F}^1_{e,f} = \{[m_e, h_f^*], [\overline{m_e}, \overline{h_f^*}], [\overline{h_f}, m_f^*]\}$ por lo que $\{[m_e, h_f^*], [\overline{m_e}, \overline{h_f^*}]\} \subset \mathcal{F}^1_{V_t^T,V_t^B}$ y $[\overline{h_f}, m_f^*] \in \mathcal{F}^1_{V_t^B}$.

Para demostrar que la fórmula $\mathcal{F}^1_{V_t^T,V_t^B}$ contiene, a lo máximo, dos cláusulas de una única fórmula elemental supongamos, por reducción al absurdo, que $\mathcal{F}_{e_1,f_1}$ ($e_1 < f_1$) es otra fórmula elemental 3SAT con $v_{e_1} \in N(v_t)$ y $v_{f_1} > v_t$, en las condiciones anteriores. Como v_e y v_{e_1} pertenecen ambas al conjunto $N(v_t)$, son adyacentes entre sí (en G) pues al ser G un grafo cordal $G(N(v_t))$ es un subgrafo inducido completo. Sin pérdida de generalidad supongamos que $v_e < v_{e_1}$. Pueden ocurrir ahora dos casos: $\mathcal{F}_{e,e_1}$ es una fórmula elemental de los tipos II a V o es una de las fórmulas elementales de los tipos VI a XII.

1. Si $\mathcal{F}_{e,e_1}$ es una fórmula de II a V, como los literales $\overline{h_{e_1}}$ y $\overline{h_e}$ están forzados por la atribución τ^0, la variable m_{e_1} está también atribuida por la propagación unitaria de la misma atribución τ^0 en $\mathcal{F}$ (v. cuadro 4.1). En particular, $\overline{h_{e_1}}, m_{e_1}^* \in \tau^1$ y, por lo tanto, ninguna de las cláusulas de $\mathcal{F}^1_{e_1,f_1}$ puede pertenecer a $\mathcal{F}^1_{V_t^T,V_t^B}$ (v. Cuadro 4.3).

2. Si $\mathcal{F}_{e,e_1}$ es una fórmula VI a XII, como la atribución del valor *verdadero* a los literales negados $\overline{h_e}$ y $\overline{h_{e_1}}$ está forzada por la atribución τ^0, la variable m_e está también atribuida por propagación unitaria (v. cuadro 4.3). Dado que $\mathcal{F}_{e,f}$ es una fórmula elemental 3SAT y $\overline{h_e}, m_e^*, \in \tau^1$, ninguna cláusula de $\mathcal{F}^1_{e,f}$ podrá pertenecer a $\mathcal{F}^1_{V_t^T,V_t^B}$.

En ambos casos tendríamos una contradicción con la hipótesis, por lo que todas las cláusulas de $\mathcal{F}^1_{V_t^T,V_t^B}$ provienen de una misma fórmula elemental. □

Lema 4.8. *Sean $C_0 \in (\mathcal{F}^E_{s,t})^T$ ($s < t$) una cláusula con 3 literales y v_π el mayor vértice del conjunto $N^*(v_s)$. Entonces para cada arista $\{v_i, v_j\} \in A$ con $v_i \in V_\pi^T$ y $v_j \in V_\pi^B$, la variable h_i está atribuida por τ^1 en (4.5).*

Demostración. Dado que $v_i < v_\pi < v_j$, $y_a^i > y_a^\pi > y_a^j > y_b^i$, el vértice $v_i \in N(v_\pi)$. Como $v_s \in N(v_\pi)$ y G es un grafo cordal, entonces $\{v_i, v_s\} \in A$.

Si $v_i \in N^*(v_s)$ $\mathcal{F}_{s,i}$ es una fórmula elemental de los tipos II a V o de los tipos IX a XII porque, en caso contrario, cada cláusula $C_0 \in (\mathcal{F}_{s,t}^E)^T$ no sería una cláusula con 3 literales según los lemas 4.4 y 4.2. Sin embargo, dado que v_π es el mayor de los vértices adyacentes a v_s, $y_a^s > y_a^i > y_a^\pi > y_b^s > y_a^j > y_b^i$, por lo que $y_a^s > y_a^i > y_b^s > y_b^i$ y, por consiguiente, $\mathcal{F}_{s,i}$ sólo podrá ser una fórmula elemental de los tipos IX a XII. Por tanto, de acuerdo al cuadro 4.3, la variable h_i está atribuida por τ^1.

Suponiendo que $v_i \in N(v_s)$, $v_i < v_s < v_j$ se tiene $y_a^i > y_a^s > y_a^\pi > y_b^s > y_a^f > y_b^i$. Por tanto, como $y_a^i > y_a^s > y_b^s > y_b^i$, $\mathcal{F}_{i,s}$ es una fórmula elemental III o IV debido a la existencia de la cláusula $C_0 \in (\mathcal{F}_{s,t}^E)^T$. Por lo que, como $m_s^* \in \tau^0$, la variable h_i está atribuida por la atribución τ^1.

Por tanto, en ambos casos, la variable h_i está atribuida por τ^1. □

Lema 4.9. *Sea $C_0 \in (\mathcal{F}_{s,t}^E)^T$ ($s < t$) una cláusula con 3 literales y v_π el mayor vértice adyacente a v_s en la partición definida en 4.6. Entonces la fórmula $\mathcal{F}^1_{V_\pi^T, V_\pi^B}$ es vacía o, como mucho, contiene dos cláusulas de una misma fórmula elemental con dos literales cada una.*

Demostración. Supongamos que $\mathcal{F}^1_{V_\pi^T, V_\pi^B} \neq \emptyset$. Entonces existe una cláusula $C_1 \in \mathcal{F}^1_{e,f}$, con $v_e \in V_\pi^T$ y $v_f \in V_\pi^B$ de modo que C_1 contiene un literal asociado al vértice v_e y otro asociado al vértice v_f. Por lo tanto, $\mathcal{F}_{e,f}$ es una fórmula elemental de uno de los tipos III, IV o IX-XII. Dado que $y_a^e > y_a^\pi > y_a^f > y_b^e$, $v_e \in N(v_\pi)$. Por construcción resulta también que $v_f > v_\pi$ y $v_f \notin N^*(v_s)$, pues v_π es el mayor de los elementos del conjunto $N^*(v_s)$. Como la variable h_e está atribuida por la atribución τ^1 (v. Lema 4.8) $\mathcal{F}_{e,f}$ es una fórmula 3SAT y $\overline{h_e} \in \tau^1$, pues en caso contrario la cláusula C_1, verdadera para la atribución τ^1, sería excluida por el algoritmo de propagación unitaria. Por tanto, en el caso de que la variable m_f no esté atribuida, $\mathcal{F}^1_{e,f} = \{[m_e, h_f^*], [\overline{m_e}, \overline{h_f^*}], [m_f^*, \overline{h_f}]\}$ por lo que se verifica que $\{[m_e, h_f^*], [\overline{m_e}, \overline{h_f^*}]\} \subset \mathcal{F}^1_{V_\pi^T, V_\pi^B}$ y que $[m_f^*, \overline{h_f}] \in \mathcal{F}^1_{V_\pi^B}$.

Probemos ahora por reducción al absurdo que la fórmula $\mathcal{F}^1_{V_\pi^T, V_\pi^B}$ contiene, a lo sumo, dos cláusulas de una única fórmula elemental. Supongamos, que $\mathcal{F}_{e_1,f_1}$ ($e_1 < f_1$) es una fórmula elemental 3SAT con $v_{e_1} \in N(v_\pi)$ y $v_{f_1} > v_\pi$, en las mismas condiciones que $\mathcal{F}_{e,f}$. Como v_e y v_{e_1} pertenecen ambas al conjunto $N(v_\pi)$, son adyacentes entre sí (en G). Supongamos, sin pérdida de generalidad, que $v_e < v_{e_1}$. De nuevo había que considerar, como anteriormente, dos casos. Pero, recordando que para ambos vértices, $\overline{h_e}, \overline{h_{e_1}} \in \tau^1$, bien sea $\mathcal{F}_{e,e_1}$ una fórmula elemental II-V, bien sea VI-XII, un de los pares de literales $\{\overline{h_e}, m_e^*\}$ o $\{\overline{h_{e_1}}, m_{e_1}^*\}$ estaría atribuido por la atribución τ^1 (de un modo similar a la demostración del Lema 4.4) por lo que $\mathcal{F}^1_{e,f} = \emptyset$ o $\mathcal{F}^1_{e_1,f_1} = \emptyset$, en contradicción con la hipótesis.

Por lo tanto, si $\mathcal{F}^1_{V_t^T, V_t^B} \neq \emptyset$ entonces todas las sus cláusulas (con dos literales cada una) provienen de una misma fórmula elemental. □

Gracias a los resultados anteriores, ahora estamos en disposición de utilizar el estudio de los vértices condicionados para probar el carácter polinomial del problema EOSC.

4.4. EOSC-SAT es subclase de 2PURL

En la presente sección demostraremos que EOSC-SAT es una subclase de 2PURL. Para ello, veremos que el algoritmo ELIMINALITERALES$(\mathcal{F}, 2)$ devuelve una fórmula conteniendo la cláusula nula si y sólo si la fórmula proposicional no es satisfacible. La condición suficiente es una simple comprobación, porque si ELIMINALITERALES$(\mathcal{F}, 2)$ contiene la cláusula nula, entonces esta fórmula no es satisfacible. Y, como $\mathcal{F}^E$ y $\mathcal{F}$ son lógicamente equivalentes, $\mathcal{F}$ tampoco es satisfacible.

Para demostrar que la condición es también necesaria consideremos una entrada del problema EOSC con respuesta negativa, el grafo de interacciones conexo $G = (V, A)$ y la correspondiente fórmula proposicional $\mathcal{F}$, que en estas condiciones no es satisfacible. Por reducción al absurdo, supongamos que la fórmula proposicional lógicamente equivalente sin literales 2p-eliminables, $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F}, 2)$, no contiene la cláusula nula.

En lo que sigue, abreviaremos PROPUNIT$(\mathcal{F}^E \cup \mathcal{U}(\tau^0))$ como $(\mathcal{F}^1, \tau^1)$ y ELIMINALITERALES$(\mathcal{F}^1)$ como $\mathcal{F}^{1,E}$.

Por recurrencia mostraremos que, bajo ciertas condiciones, si existe un vértice v_π tal que en la partición (4.6) de la fórmula $\mathcal{F}^E$, $\mathcal{F}^E_{V_\pi^B}$ es una fórmula satisfacible, entonces existe un vértice $v_\phi < v_\pi$ para el cual $\mathcal{F}^E_{V_\phi^B}$ es también una fórmula satisfacible. Para el referido vértice v_π, considerando una atribución de verdad τ^0 convenientemente escogida, la existencia del vértice v_ϕ está asociada a la existencia de una cláusula con tres literales en $\mathcal{F}^{1,E}_{V_\pi^T}$, bajo condiciones que veremos posteriormente. Esto permite identificar una secuencia de vértices de V, finita, $v_{\pi_1} > \cdots > v_{\pi_t}$ (pues V es un conjunto finito), ilustrada en la Figura 4.9 (en la cual los vértices se suponen en orden creciente empezando por arriba). Para el vértice v_{π_t}, la fórmula $\mathcal{F}^{1,E}_{V_{\pi_t}^T}$ no contiene cláusulas con 3 literales.

A lo largo del estudio constataremos que v_ϕ es un vértice no condicionado o que es adyacente (maximal) a un vértice no condicionado.

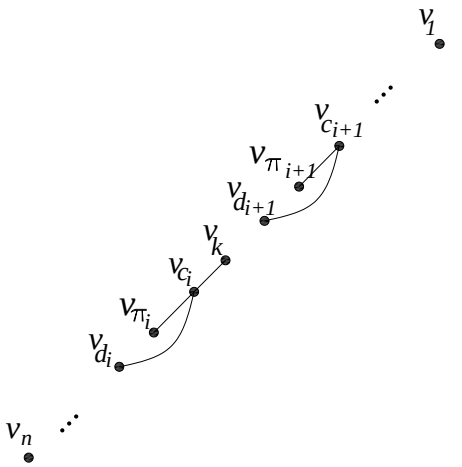


Figura 4.9: Secuencia de vértices del grafo de interacciones de una instancia de EOSC, $v_{\pi_1} > \cdots > v_{\pi_t}$, que induce una partición en la correspondiente fórmula proposicional $\mathcal{F}^E$.

En estas condiciones, veremos que existe un modelo para $\mathcal{F}^E$, y por lo tanto para $\mathcal{F}$, en contradicción con la hipótesis que habíamos considerado de ser $\mathcal{F}$ una fórmula no satisfacible. Por tanto, si $\mathcal{F}$ no es satisfacible, la fórmula retornada por el algoritmo ELIMINALITERALES$(\mathcal{F}, 2)$ contiene la cláusula nula.

La secuencia anterior contiene por lo menos un vértice v_{π_1}. A este efecto obsérvese que la fórmula proposicional $\mathcal{F}^E$ contiene una cláusula con 3 literales. Si (por reducción al absurdo) $\mathcal{F}^E$ fuese una instancia 2SAT, de acuerdo al Teorema 2.8 contendría una cláusula nula, lo que estaría en contradicción con la hipótesis. De todas las fórmulas elementales $\mathcal{F}^E_{s,t}$ $(s < t)$, conteniendo cláusulas con 3 literales, consideremos una cuyo valor de t sea máximo. Denotemos $\mathcal{F}^E_{s_1,t_1}$ a esa fórmula elemental y sea C_1 una cláusula con tres literales. Pueden darse dos casos distintos: $C_1 \in (\mathcal{F}^E_{s_1,t_1})^B$ o $C_1 \in (\mathcal{F}^E_{s_1,t_1})^T$. El vértice v_{π_1} a considerar depende de cada uno de estos casos que estudiamos separadamente a continuación.

1. $C_1 \in (\mathcal{F}^E_{s_1,t_1})^B$

   Suponiendo que C_1 es una cláusula con 3 literales en $(\mathcal{F}^E_{s_1,t_1})^B$, se tiene que $C_1 = [\overline{h_{t_1}}, m^*_{t_1}, h^*_{s_1}]$ (v. Cuadro 4.1). Como consecuencia de los lemas 4.3 y 4.4, para cada uno de los vértices $v_i \in N(v_{t_1})$ $\{v_i, v_{t_1}\}$ es una arista de propagación, por lo que v_{t_1} es un vértice no condicionado.

   Tomando como referencia el vértice v_{t_1} considérese la partición de la fórmula proposicional $\mathcal{F}^E = \mathcal{F}^E_{V^B_{t_1}} \cup \mathcal{F}^E_{V^T_{t_1}} \cup \mathcal{F}^E_{V^T_{t_1},V^B_{t_1}}$ en las mismas

condiciones que en (4.6), i.e. $V_{t_1}^T = \{v_i \in V_0 : v_i \leq v_{t_1}\}$, $V_{t_1}^B = \{v_i \in V_0 : v_i > v_{t_1}\}$. Entonces $\mathcal{F}^E_{V_{t_1}^B}$ y $\mathcal{F}^E_{V_{t_1}^T}$ son fórmulas variable disjuntas. La fórmula $\mathcal{F}^E_{V_{t_1}^T,V_{t_1}^B}$ está formada por las cláusulas de $\mathcal{F}^E$ conteniendo, simultáneamente, un literal sobre una variable asociada a un vértice de cada uno de los dos conjuntos $V_{t_1}^B$ e $V_{t_1}^T$. Por construcción, cada una de las fórmulas $\mathcal{F}^E_{i,j}$ $(i < j)$ con $v_j > v_{t_1}$ no contiene cláusulas con 3 literales por lo que $\mathcal{F}^E_{V_{t_1}^B}$ es una instancia 2SAT sin literales p-eliminables.

Considérese ahora la atribución de verdad $\tau^{0,1} = \{h_{t_1}, m^*_{t_1}\}$ ($\tau^{0,1} \subset \overline{Flip(C_1, m^*_{t_1})}$), su propagación unitaria en $\mathcal{F}^E$ definida en (4.5) y la partición de esta fórmula indicada en (4.6) con $v_{\pi_1} = v_{t_1}$. Dado que $\mathcal{F}^E$ no contiene literales p-eliminables, la fórmula $\mathcal{F}^1$ no contiene a la cláusula nula y, de acuerdo al Lema 4.7, $\mathcal{F}^1_{V_{t_1}^T,V_{t_1}^B} = \emptyset$ o $\mathcal{F}^1_{V_{t_1}^T,V_{t_1}^B} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$ para una arista $\{v_{c_1}, v_{d_1}\}$ con $v_{c_1} \in V_{t_1}^T$ y $v_{d_1} \in V_{t_1}^B$.

Excluyendo todos los literales p-eliminables en la fórmula $\mathcal{F}^1$ se obtiene $\mathcal{F}^{1,E} = \text{ELIMINALITERALES}(\mathcal{F}^1)$, la cual no contiene la cláusula nula porque, por hipótesis, $\mathcal{F}^E$ no contiene literales 2p-eliminables. Por lo tanto, en la partición $\mathcal{F}^{1,E} = \mathcal{F}^{1,E}_{V_{t_1}^B} \cup \mathcal{F}^{1,E}_{V_{t_1}^T} \cup \mathcal{F}^{1,E}_{V_{t_1}^T,V_{t_1}^B}$ se verifica o $\mathcal{F}^{1,E}_{V_{t_1}^T,V_{t_1}^B} = \emptyset$ o $\mathcal{F}^{1,E}_{V_{t_1}^T,V_{t_1}^B} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$ y, en este caso, $\mathcal{F}^{1,E}_{V_{t_1}^B}$ es una fórmula 2SAT sin literales p-eliminables y por tanto satisfacible.

Si $\mathcal{F}^{1,E}_{V_{t_1}^T,V_{t_1}^B} = \emptyset$, como las fórmulas $\mathcal{F}^{1,E}_{V_{t_1}^B}$ y $\mathcal{F}^{1,E}_{V_{t_1}^T}$ no tienen variables en común, $\mathcal{F}^{1,E}$ no es satisfacible y $\mathcal{F}^{1,E}_{V_{t_1}^B}$ es satisfacible entonces la fórmula $\mathcal{F}^{1,E}_{V_{t_1}^T}$ es una fórmula no satisfacible sin literales p-eliminables por lo que contiene una cláusula con 3 literales.

Supongamos ahora que $\mathcal{F}^{1,E}_{V_{t_1}^T,V_{t_1}^B} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$. Dado que la fórmula $\mathcal{F}^{1,E}_{V_{t_1}^B}$ es una instancia 2SAT, la fórmula $\mathcal{F}^{1,E}_{V_{t_1}^T}$ contiene una cláusula con 3 literales. En caso contrario, $\mathcal{F}^{1,E}$ sería una instancia 2SAT, sin literales p-eliminables y por tanto satisfacible. Por lo que la atribución de verdad τ^1 admite un prolongamiento (en el conjunto de los literales de $\mathcal{F}$) satisfaciendo la fórmula proposicional $\mathcal{F}^E$, en contradicción con la hipótesis de partida.

2. $C_1 \in (\mathcal{F}^E_{s_1,t_1})^T$

Suponiendo que $C_1 \in (\mathcal{F}^E_{s_1,t_1})^T$ entonces $C_1 = [h_{s_1}, m^*_{s_1}, h^*_{t_1}]$, y, de acuerdo al Lema 4.2, para cada vértice $v_j \in N^*(v_{s_1})$, $\mathcal{F}_{s_1,j}$ es una

fórmula elemental II a V (II –V) o IX–XII. Designemos por v_{π_1} el mayor vértice adyacente a v_{s_1}. Tal vértice existe pues $v_{t_1} \in N^*(v_{s_1})$ por lo que $v_{\pi_1} \geq v_{t_1}$.

De un modo similar al caso 1, anterior, tomando el vértice v_{π_1} como referencia, considérese la partición $\mathcal{F}^E = \mathcal{F}^E_{V^B_{\pi_1}} \cup \mathcal{F}^E_{V^T_{\pi_1}} \cup \mathcal{F}^E_{V^T_{\pi_1},V^B_{\pi_1}}$. También en este caso $\mathcal{F}^E_{V^B_{\pi_1}}$ es una fórmula proposicional 2SAT. Considérese ahora la atribución de verdad $\tau^{0,1} = \{\overline{h_{s_1}}, m^*_{s_1}\}$ ($\tau^{0,1} \subset \overline{Flip(C_1, m^*_{s_1})}$) y su propagación unitaria en $\mathcal{F}^E$ cf. la ecuación (4.5) y la partición (4.6). Como, por hipótesis, $\mathcal{F}^E$ no contiene literales 2p-eliminables, $\mathcal{F}^1$ no contiene a la cláusula nula y de acuerdo al Lema 4.9, $\mathcal{F}^1_{V^T_{\pi_1},V^B_{\pi_1}} = \emptyset$ o $\mathcal{F}^1_{V^T_{\pi_1},V^B_{\pi_1}} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$ para alguna arista $\{v_{c_1}, v_{d_1}\}$ con $v_{c_1} \in V^T_{\pi_1}$ y $v_{d_1} \in V^B_{\pi_1}$.

Excluyendo los literales p-eliminables en $\mathcal{F}^1$, la fórmula proposicional $\mathcal{F}^{1,E} = \text{ELIMINALITERALES}(\mathcal{F}^1)$ no contiene la cláusula nula pues, por hipótesis, $\mathcal{F}^E$ no contiene literales 2p-eliminables. Por lo que, considerando la partición $\mathcal{F}^{1,E} = \mathcal{F}^{1,E}_{V^B_{\pi_1}} \cup \mathcal{F}^{1,E}_{V^T_{\pi_1}} \cup \mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}}$, la fórmula $\mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}} = \emptyset$ o $\mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$ y entonces $\mathcal{F}^{1,E}_{V^B_{\pi_1}}$ es una fórmula 2SAT.

Si $\mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}} = \emptyset$, como $\mathcal{F}^{1,E}$ no es satisfacible, entonces $\mathcal{F}^{1,E}_{V^T_{\pi_1}}$ tampoco es satisfacible, pues $\mathcal{F}^{1,E}_{V^B_{\pi_1}}$ y $\mathcal{F}^{1,E}_{V^T_{\pi_1}}$ son variable-disjuntas y la primera es satisfacible, por lo que contiene una cláusula con 3 literales.

Suponiendo que $\mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$, $\mathcal{F}^{1,E}_{V^T_{\pi_1}}$ igualmente contiene una cláusula con 3 literales, caso contrario $\mathcal{F}^{1,E}$ (una instancia 2SAT, sin literales p-eliminables) sería una fórmula satisfacible y, por consiguiente, $\mathcal{F}$ sería satisfacible, contrariamente a la hipótesis admitida.

Del análisis de los puntos anteriores, se concluye que, para el vértice v_{π_1} se tiene:

- existe $C_1 \in \mathcal{F}^E_{s_1,t_1}$ ($s_1 < t_1$), $|C_1| = 3$ y v_{t_1} es máximo;
- $\mathcal{F}^E_{V^B_{\pi_1}}$ es 2SAT, satisfacible;
- $\mathcal{F}^{1,E}_{V^T_{\pi_1}}$ contiene una cláusula con 3 literales;
- $\mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}} = \emptyset \vee \mathcal{F}^{1,E}_{V^T_{\pi_1},V^B_{\pi_1}} = \{[m_{c_1}, h^*_{d_1}], [\overline{m_{c_1}}, \overline{h^*_{d_1}}]\}$, $v_{c_1} \in V^T_{\pi_1}$ y $v_{d_1} \in V^B_{\pi_1}$;
- $\mathcal{F}^{1,E}_{V^B_{\pi_1}}$ es satisfacible y $\mathcal{F}^{1,E}_{V^T_{\pi_1}}$ no.

Por inducción, admitamos que existe una secuencia de vértices $v_{\pi_1}, \cdots, v_{\pi_i}$ de modo que para cada v_{π_i}, existen una cláusula C_i y una atribución de verdad $\tau^{0,i}$, elegida de acuerdo con esta cláusula, conforme a que $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$ o $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$, en las siguientes condiciones:

1. $C_i \in \mathcal{F}^E_{s_i,t_i}$, $(s_i < t_i)$, $|C_i| = 3$ y v_{t_i} máximo en $V^T_{\pi_{i-1}}$;
2. $(\mathcal{F}^i, \tau^i) = PropUnit(\mathcal{F}^E \cup \mathcal{U}(\tau^{0,i}))$;
3. $\tau^{0,i} = \{\overline{h_{s_i}}, m^*_{s_i}\}$ o $\tau^{0,i} = \{h_{t_i}, m^*_{t_i}\}$, respectivamente;
4. $\mathcal{F}^{i,E} = EliminaLiterales(\mathcal{F}^i)$;
5. $\mathcal{F}^{i,E} = \mathcal{F}^{i,E}_{V^B_{\pi_i}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_i}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_i},V^B_{\pi_i}}$
6. $\mathcal{F}^{i,E}_{V^B_{\pi_i}}$ y $\mathcal{F}^{i,E}_{V^T_{\pi_i}}$ son variable disjuntas;
7. $\mathcal{F}^{i,E}_{V^T_{\pi_i},V^B_{\pi_i}} = \emptyset \vee \mathcal{F}^{i,E}_{V^T_{\pi_i},V^B_{\pi_i}} = \{[m_{c_i}, h^*_{d_i}], [\overline{m_{c_i}}, \overline{h^*_{d_i}}]\}$
8. $v_{c_i} \in V^T_{\pi_i}$ y $v_{d_i} \in V^B_{\pi_i}$ (v. Figura 4.9);
9. $\mathcal{F}^{i,E}_{V^B_{\pi_i}}$ es satisfacible y $\mathcal{F}^{i,E}_{V^T_{\pi_i}}$ no.

Como $\mathcal{F}^{i,E}_{V^T_{\pi_i}}$ no es satisfacible, contiene una cláusula con 3 literales, entonces existe una fórmula elemental $\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}}$ $(s_{i+1} < t_{i+1})$, con $v_{t_{i+1}}$ máximo en $V^T_{\pi_i}$, conteniendo una cláusula C_{i+1} con 3 literales. (e con $v_{t_{i+1}} \in V^T_\pi$). Por lo tanto $\mathcal{F}_{s_{i+1},t_{i+1}}$ es una fórmula elemental de los tipos IX a XII, pudiendo darse de nuevo dos casos: $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^B$ o $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^T$.

Análogamente a los casos anteriores, para $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^B$, $C_{i+1} = [\overline{h_{t_{i+1}}}, m^*_{t_{i+1}}, h^*_{s_{i+1}}]$ por lo que que consideraremos

$$v_{\pi_{i+1}} = v_{t+1} \text{ y } \tau^{0,i+1} = \{h_{t_{i+1}}, m^*_{t_{i+1}}\} \subset \overline{Flip(C_{i+1}, m^*_{t_{i+1}})}.$$

Y, para $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^T$, $C_{i+1} = [h_{s_{i+1}}, m^*_{s_{i+1}}, h^*_{t_{i+1}}]$ por lo que asumiremos $v_{\pi_{i+1}}$ como el mayor de los vértices adyacentes a $v_{s_{i+1}}$. Este vértice existe pues $v_{t_{i+1}} \in N^*(v_{s_{i+1}})$. Para la atribución de verdad consideremos

$$\tau^{0,i+1} = \{\overline{h_{s_{i+1}}}, m^*_{s_{i+1}}]\} \subset \overline{Flip(C_{i+1}, m^*_{s_{i+1}})}.$$

Llamando al conjunto $V_{\pi_{i+1},\pi_i} = \{v_i \in V : v_{\pi_{i+1}} < v_i \leq v_{\pi_i}\}$, considérese la partición

$$\mathcal{F}^{i,E} = \mathcal{F}^{i,E}_{V^B_{\pi_i}} \cup \mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_{i+1}}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_{i+1}},V_{\pi_{i+1},\pi_i}} \cup \mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i},V^B_{\pi_i}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_{i+1}},V^B_{\pi_i}} \tag{4.7}$$

Es conocido que v_{π_i} resulta de la existencia de una cláusula $C_i \in \mathcal{F}^E_{s_i,t_i}$ con tres literales y que $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$ o $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$. Combinando cada uno de estos dos casos con el hecho de que $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^B$ o $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^T$, se tiene la necesidad de analizar cuatro casos. Para el estudio de los dos primeros, consideremos $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$, por lo que $v_{\pi_i} = v_{t_i}$ y $\tau^{0,i} = \{h_{t_i}, m^*_{t_i}\}$.

A.1 $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$ y $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^B$

Recordemos que, de acuerdo al Lema 4.6, para cada $v_i \in N(v_{t_i})$ la variable h_i queda atribuida por la atribución de verdad τ^i. Por lo tanto el vértice $v_{t_{i+1}} \notin N(v_{t_i})$. Luego, como $v_{\pi_{i+1}} = v_{t_{i+1}}$ y $v_{\pi_i} = v_{t_i}$, se concluye que $v_{\pi_{i+1}} < v_{\pi_i}$ (y en el presente caso $v_{\pi_{i+1}} \notin N(v_{\pi_i})$).

Probemos ahora que la fórmula $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ es una instancia 2SAT. A este efecto consideremos una fórmula elemental $\mathcal{F}^{i,E}_{i,j}$, con $i < j$. Si $v_j = v_{\pi_i}$ y dado que $v_{\pi_i} = v_{t_i}$ y que $\{v_i, v_j\}$ es una arista de propagación, entonces $\mathcal{F}^{i,E}_{i,j} = \emptyset$. En el caso de que $v_{\pi_i} > v_j > v_{\pi_{i+1}}$, como $v_{\pi_{i+1}} = v_{t_{i+1}}$, debido a la maximalidad en la elección del vértice $v_{t_{i+1}}$, $\mathcal{F}^{i,E}_{i,j}$ no contiene cláusulas con 3 literales. Lo que permite concluir que, efectivamente, $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ es una instancia 2SAT.

En $\mathcal{F}$ no hay ninguna fórmula elemental $\mathcal{F}_{\alpha,\beta}$ con $v_\alpha \in V^T_{t_{i+1}}$ y $v_\beta \in V^B_{t_i}$. Si la hubiera, $v_\alpha \in N(v_{t_i})$ y $v_\alpha \in N(v_{t_{i+1}})$, por lo que $\mathcal{F}_{\alpha,t_i}$ sería una fórmula elemental III, IV o 3SAT (IX–XII) y, por tanto, la variable h_α estaría atribuida por la atribución τ^i. Como $v_{t_{i+1}}$ es un vértice no condicionado, en el caso de que $\mathcal{F}_{\alpha,t_{i+1}}$ fuese una fórmula III o IV, la variable $m_{t_{i+1}}$ estaría igualmente atribuida por τ^i por lo que $|C_{i+1}| \leq 2$. Si fuese una fórmula elemental IX a XII, como ni $h_{t_{i+1}}$ ni $m_{t_{i+1}}$ están atribuidas entonces la cláusula $[\overline{h_{t_{i+1}}}, m^*_{t_{i+1}}] \in \mathcal{F}^i_{\alpha,t_{i+1}}$. Por lo tanto $|C_{i+1}| \leq 2$. En cualquiera de los casos entramos en contradicción con la existencia de la cláusula $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^B$ con 3 literales distintos. Por lo tanto, en la partición (4.7) la fórmula $\mathcal{F}^{i,E}_{V^T_{\pi_{i+1}},V^B_{\pi_i}} = \emptyset$.

A.2 $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$ y $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^T$

Dado que $C_{i+1} = [\overline{h_{t_{i+1}}}, m^*_{t_{i+1}}, h^*_{s_{i+1}}]$, ninguna de las variables $h_{s_{i+1}}$ y $h_{t_{i+1}}$ está atribuida por τ^i. Por tanto $v_{t_{i+1}} \notin N(v_{\pi_i})$ (en este caso $v_{\pi_i} = v_{t_i}$), por lo que $y_b^{s_{i+1}}, y_b^{t_{i+1}} > y_a^{t_i}$. El vértice $v_{\pi_{i+1}}$ tampoco es adyacente a $v_{t_i} = v_{\pi_i}$. En caso contrario, $\{v_{\pi_{i+1}}, v_{t_i}\}$ sería una arista de propagación por lo que $y_a^{\pi_{i+1}} > y_a^{t_i} > y_b^{\pi_{i+1}}$. Por otro lado, dado que $v_{s_{i+1}}, v_{t_{i+1}}$ y $v_{\pi_{i+1}}$ son adyacentes entre sí,

$y_a^{s_{i+1}} > y_a^{\pi_{i+1}} > y_b^{s_{i+1}} > y_a^{t_i} > y_b^{\pi_{i+1}}$ y, por consiguiente, $\mathcal{F}_{s_{i+1},\pi_{i+1}}$ sería una fórmula elemental de uno de los tipos VI a XII. Combinando este hecho con la hipótesis de que $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^T$, $\mathcal{F}_{s_{i+1},\pi_{i+1}}$ tendría que ser una fórmula 3SAT. Dado que $h_{\pi_{i+1}}$ estaría atribuida por la atribución τ^i, la cláusula $[h_{s_{i+1}}, m^*_{s_{i+1}}] \in \mathcal{F}^1_{s_{i+1},\pi_{i+1}}$ pues ni la variable $h_{s_{i+1}}$ ni $m_{s_{i+1}}$ están atribuidas por τ^i. Pero, en este caso, ninguna de las cláusulas de $(\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^T$ podría contener tres literales, en contradicción con la existencia de la cláusula C_{i+1}. Por lo que $v_{\pi_{i+1}} < v_{\pi_i}$ (y también en este caso $v_{\pi_{i+1}} \notin N(v_{\pi_i})$).

Probamos ahora que $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ es una instancia 2SAT. Consideramos una fórmula elemental $\mathcal{F}^{i,E}_{i,j}$, con $i < j$, se $v_j = v_{\pi_i}$, como hicimos en el caso anterior, $\mathcal{F}^{i,E}_{i,j} = \emptyset$. Suponiendo que $v_{\pi_i} > v_j > v_{\pi_{i+1}} \geq v_{t_{i+1}}$, por construcción $\mathcal{F}^{i,E}_{i,j}$ no contiene cláusulas con 3 literales, atendiendo a la elección del vértice $v_{t_{i+1}}$.

El grafo de interacciones no contiene ninguna arista $\{v_\alpha, v_\beta\}$ con $v_\alpha \in V^T_{\pi_{i+1}}$ y $v_\beta \in V^B_{\pi_i}$. Si la hubiera, $v_\alpha \in N(v_{\pi_i})$ por lo que la variable h_α estaría atribuida por la atribución τ^i. Por otro lado, $v_\alpha \in N(v_{\pi_{i+1}})$ por lo que $\{v_\alpha, v_{s_{i+1}}\} \in A$. Por tanto, $v_\alpha \in N^*(v_{s_{i+1}})$ o $v_\alpha \in N(v_{s_{i+1}})$. En el primer caso, $\mathcal{F}_{s_{i+1},\alpha}$ sería una fórmula elemental de uno de los tipos VI a XII. Debido a que la variable h_α estaría atribuida por τ^i, bien $[h_{s_{i+1}}, m^*_{s_{i+1}}] \in \mathcal{F}^i$, bien una de las variables $h_{s_{i+1}}$ o $m_{s_{i+1}}$ estaría atribuida. En cualquiera de los casos se entra en contradicción con la existencia de la cláusula C_{i+1}. Si fuese $v_\alpha \in N(v_{s_{i+1}})$, $\mathcal{F}_{\alpha,s_{i+1}}$ sería una fórmula de uno de los tipos II a V, por lo que la variable $m_{s_{i+1}}$ estaría atribuida por τ^i, lo que de nuevo entraría en contradicción con la existencia de C_{i+1} con 3 literales distintos.

Concluido el estudio de los dos primeros casos, pasemos a la segunda parte considerando ahora que $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$, con $|C_i| = 3$. Recordemos que $C_i = [h_{s_i}, m^*_{s_i}, h^*_{t_i}]$, que v_{π_i} es el mayor vértice del conjunto $N^*(v_{s_i})$ y que $\tau^{0,i} = \{\overline{h_{s_i}}, m^*_{s_i}\}$

Suponiendo que $\mathcal{F}^{i,E}_{V^T_{\pi_i}}$ contiene una cláusula con tres literales, consideremos $\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}}$ ($s_{i+1} < t_{i+1}$) una fórmula elemental conteniendo una cláusula C_{i+1} con 3 literales, con $v_{t_{i+1}} \in V^T_{\pi_i}$ máximo. El vértice $v_{t_{i+1}} \notin N^*(v_{s_i})$ pues, en el caso contrario $\mathcal{F}_{s_i,t_{i+1}}$ sería una fórmula elemental de uno de los tipos II a V o IX a XII (v. Lema 4.2). Si fuese una fórmula IX-XII, $h_{t_{i+1}}$ estaría atribuida por la atribución τ^i por lo que que $|C_{i+1}| \leq 2$, en contradicción con la hipótesis. Si fuese una fórmula II-V, la variable $m_{t_{i+1}}$ estaría atribuida por la atribución τ^i por lo que $C_{i+1} \in (\mathcal{F}^i_{s_{i+1},t_{i+1}})^T$. Pero como $v_{t_{i+1}} \in N^*(v_{s_i})$, entonces $\{v_{s_{i+1}}, v_{s_i}\} \in A$.

Por lo que bien una de las variables $h_{s_{i+1}}$ o $m_{s_{i+1}}$ está atribuida por τ^i, o bien $[h_{s_{i+1}}, m^*_{s_{i+1}}] \in \mathcal{F}^i$. En cualquiera de los casos estamos en contradicción con la existencia de la cláusula $C_{i+1} \in (\mathcal{F}^i_{s_{i+1},t_{i+1}})^T$ con 3 literales. En estas condiciones consideraremos a continuación, por separado, cada uno de los dos casos $C_{i+1} \in (\mathcal{F}^i_{s_{i+1},t_{i+1}})^B$ y $C_{i+1} \in (\mathcal{F}^i_{s_{i+1},t_{i+1}})^T$.

B.1 $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$ y $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^B$

El vértice $v_{t_{i+1}} < v_{s_i}$, pues, en caso contrario, $v_{t_{i+1}}$ estaría entre los vértices v_{π_i} y v_{s_i} y, por tanto, $v_{t_{i+1}} \in N^*(v_{s_i})$. Uno de los literales sobre una de las variables $h_{t_{i+1}}, m_{t_{i+1}}$ estaría atribuido por la atribución τ^i, y, por consiguiente, cualquiera de las cláusulas de $(\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^B$ tendría a lo sumo 2 literales. Lo que está en contradicción con la hipótesis considerada, $|C_{i+1}| = 3$. Como $v_{\pi_i} > v_{s_i} > v_{t_{i+1}}$ y, en este caso, $v_{t_{i+1}} = v_{\pi_{i+1}}$, se concluye que $v_{\pi_{i+1}} < v_{\pi_i}$.

Probemos ahora que la fórmula $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ es una instancia 2SAT. Consideremos $\mathcal{F}^{i,E}_{i,j}$ $(i < j)$ una fórmula elemental. Si $v_j = v_\pi$, tenemos que $v_j \in N^*(v_{s_i})$. Por tanto por lo menos uno de los literales de una de las variables h_j y m_j está atribuido por τ^i. En el caso de que $v_i > v_{s_i}$, uno de los literales sobre una de las variables h_i y m_i está atribuido pelo que cada cláusula de $\mathcal{F}^{i,E}_{i,j}$ tiene como máximo dos literales. En el caso contrario, si $v_i < v_{s_i}$ y $\mathcal{F}_{i,j}$ fuese una fórmula de los tipos IX a XII, $\mathcal{F}_{i,s_i}$ es una fórmula elemental de los tipos VI a XII. En cualquiera de ambos casos, dado que el literal afirmado h_{s_i} está atribuido por τ^i, la fórmula elemental $\mathcal{F}^1_{i,s_i}$ contiene la cláusula $[h_i, m^*_i]$. Considerando también que uno de los literales sobre una de las variables h_j y m_j está atribuido, resulta que $\mathcal{F}^{i,E}_{i,j}$contiene a lo sumo dos literales por cláusula.

Suponiendo que $v_{\pi_i} > v_j > v_{\pi_{i+1}} = v_{t_{i+1}}$, la fórmula elemental $\mathcal{F}^{i,E}_{i,j}$ no contiene cláusulas con 3 literales pues $v_{t_{i+1}}$ es el mayor vértice del conjunto $V^T_{\pi_i}$ en estas condiciones.

También en este caso el grafo de interacciones no contiene cualquiera arista $\{v_\alpha, v_\beta\}$ con $v_\alpha \in V^T_{t_{i+1}}$ y $v_\beta \in V^B_{\pi_i}$. Caso contrario $\mathcal{F}_{\alpha,s_i}$ sería una fórmula elemental de los tipos II a V, pues como $y^\alpha_a > y^{s_i}_a > y^\pi_a > y^{s_i}_b > y^\beta_a > y^\alpha_b$, se tiene $y^\alpha_a > y^{s_i}_a > y^{s_i}_b > y^\alpha_b$. Pero, dado que $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$, la variable m_{s_i} no podría estar atribuida por τ^i, por lo que $\mathcal{F}_{\alpha,s_i}$ solo podría ser una fórmula III o IV. Entonces la variable h_α estaría atribuida por τ^i. Por otro lado como $v_{t_{i+1}} \notin N^*(v_{s_i})$ y $v_{t_{i+1}} < v_{s_i}$ se tiene que $y^\alpha_a > y^{t_{i+1}}_a > y^{t_{i+1}}_b > y^\alpha_b$ por lo que $\mathcal{F}_{\alpha,t_{i+1}}$ es una fórmula de los tipos II a V. Pero como h_α está atribuida por τ^i, la variable $m_{t_{i+1}}$ está también atribuida por la misma atribución, en contradicción con la existencia de la cláusula $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^B$.

B.2 $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$ y $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^T$

El vértice $v_{s_{i+1}} \notin N(v_{s_i})$. Caso contrario $\mathcal{F}_{s_{i+1},s_i}$ sería una fórmula elemental (en $\mathcal{F}$) de los tipos:

- II o V, por lo que existiría una cláusula unitaria $[m^*_{s_i}] \in \mathcal{F}_{s_{i+1},s_i}$ y, por tanto, la cláusula $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$ tendría a lo sumo 2 literales;
- III o IV, por lo que la variable $h_{s_{i+1}}$ estaría atribuida pela atribución τ^i y, por lo tanto, la cláusula $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^T$ tendría como mucho 2 literales;
- de VI a XII, como el literal $h_{s_i} \in \tau^i$, la cláusula $[h_{s_{i+1}}, m^*_{s_{i+1}}] \in \mathcal{F}^i_{s_{i+1},t_{i+1}}$, por lo que cualquier cláusula en $(\mathcal{F}^{i,E}_{s_{i+1}}, v_{t_{i+1}})^T$ no podría contener 3 literales (v. Lema 4.2 considerando la existencia de la cláusula $[h_{s_{i+1}}, m^*_{s_{i+1}}]$).

En cualquiera de las tres posibilidades la existencia de la fórmula elemental considerada conduciría a una contradicción con la hipótesis.

Dado que $v_{\pi_{i+1}}$ es el mayor vértice adyacente a $v_{s_{i+1}}$, $y_a^{s_{i+1}} > y_a^{\pi_{i+1}} > y_a^{s_i} > y_a^{\pi_i}$. Por lo tanto, $v_{\pi_{i+1}} < v_{\pi_i}$. (Como nota refiramos que también en este caso $v_{\pi_{i+1}} \notin N(v_{\pi_i})$)

La fórmula $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ es una instancia 2SAT. La verificación de esta propiedad sigue la misma construcción que el caso anterior atendiendo que en este caso $v_{\pi_i} > v_j > v_{\pi_{i+1}} \geq v_{t_{i+1}}$.

G no contiene cualquiera arista $\{v_\alpha, v_\beta\}$ con $v_\alpha \in V^T_{\pi_{i+1}}$ y $v_\beta \in V^B_\pi$. Caso contrario v_α seria adyacente a v_{s_i} y $\mathcal{F}_{\alpha,s_i}$ sólo podría ser una fórmula de los tipos II a V. Sin embargo, dado que $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$, esta no puede ser una fórmula ni II ni V pues en ese caso existiría una cláusula unitaria $[m^*_{s_i}] \in \mathcal{F}$ lo que sería incompatible con la existencia de C_i. Por tanto $\mathcal{F}_{\alpha,s_i}$ es una fórmula III o IV y la variable h_α está atribuida por la atribución τ^i. Por otro lado, v_α y $v_{s_{i+1}}$ serían igualmente adyacentes en G por lo que si $v_{s_{i+1}} < v_\alpha$, $\mathcal{F}_{s_{i+1},\alpha}$ sería una fórmula de los tipos VI a XII o, si $v_\alpha < v_{s_{i+1}}$ una fórmula de los tipos II a IV. Pero si $\mathcal{F}_{s_{i+1},\alpha}$ fuese una fórmula VI a XII, como h_α está atribuida por τ^i, o $h_{s_{i+1}}$ o $m_{s_{i+1}}$ estaría igualmente atribuida por la misma atribución o la cláusula $[h_{s_{i+1}}, m^*_{s_{i+1}}] \in \mathcal{F}^i$. En cualquiera de ambos casos, entramos en contradicción con la existencia de la cláusula $C_{i+1} \in (\mathcal{F}^{i,E}_{s_{i+1},t_{i+1}})^T$. Si $\mathcal{F}_{\alpha,t_{i+1}}$ fuese una fórmula de los tipos II a V, la variable $m_{s_{i+1}}$ estaría atribuida por τ^i. Por lo tanto entramos nuevamente en contradicción con la existencia de la cláusula C_{i+1} en las condiciones de la hipótesis.

Del análisis de los cuatro casos anteriores, admitiendo que $\mathcal{F}^E$ es una fórmula sin literales 2p-eliminables, no satisfacible y que se verifican las condiciones (1) a (9) en la página 136:

1. $\mathcal{F}^{i,E}_{V^T_{\pi_i}}$ contiene una cláusula con 3 literales (distintos),
2. existe una cláusula $C_{i+1} \in \mathcal{F}^E_{s_{i+1},t_{i+1}}$ ($s_{i+1} < t_{i+1}$);
3. $|C_{i+1}| = 3$ y $v_{t_{i+1}}$ máximo en $V^T_{\pi_i}$;
4. existe un vértice $v_{\pi_{i+1}} < v_{\pi_i}$;
5. $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ es 2SAT, satisfacible;
6. $\mathcal{F}^{i,E}_{V^T_{\pi_{i+1}},V^B_{\pi_i}} = \emptyset$.

Por lo que, la partición (4.7) equivale a:

$$\mathcal{F}^{i,E} = \mathcal{F}^{i,E}_{V^B_{\pi_i}} \cup \mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_{i+1}}} \cup \mathcal{F}^{i,E}_{V^T_{\pi_{i+1}},V_{\pi_{i+1},\pi_i}} \cup \mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i},V^B_{\pi_i}} \tag{4.8}$$

Consideremos la propagación de la atribución de verdad $\tau^{0,i+1}$ en $\mathcal{F}^{i,E}$, $(\mathcal{F}^{i,i+1}, \tau^{i,i+1}) = PropUnit(\mathcal{F}^{i,E} \cup \mathcal{U}(\tau^{0,i+1}))$ donde $\tau^{0,i+1} = \{h_{t_{i+1}}, m^*_{t_{i+1}}\}$ o $\tau^{0,i+1} = \{\overline{h_{s_{i+1}}}, m^*_{s_{i+1}}\}$ según que C_{i+1} sea una cláusula en $\mathcal{F}^B_{s_{i+1},t_{i+1}}$ o $\mathcal{F}^T_{s_{i+1},t_{i+1}}$, respectivamente.

Esta fórmula proposicional, $\mathcal{F}^{i,i+1}$, no contiene la cláusula nula pues $\mathcal{F}^{i,E}$ no contiene literales p-eliminables. Y, en estas condiciones, para una partición de la fórmula $\mathcal{F}^{i,i+1}$, idéntica a (4.8):

$$\mathcal{F}^{i,i+1} = \mathcal{F}^{i,i+1}_{V^T_{\pi_{i+1}}} \cup \mathcal{F}^{i,i+1}_{V_{\pi_{i+1},\pi_i}} \cup \mathcal{F}^{i,i+1}_{V^B_{\pi_i}} \cup \mathcal{F}^{i,i+1}_{V^T_{\pi_{i+1}},V_{\pi_{i+1},\pi_i}} \cup \mathcal{F}^{i,i+1}_{V_{\pi_{i+1},\pi_i},V^B_{\pi_i}} \tag{4.9}$$

para la cual

- $\mathcal{F}^{i,i+1}_{V_{\pi_{i+1},\pi_i}}$ es una instancia 2SAT;
- $\mathcal{F}^{i,i+1}_{V_{t_{i+1},t_i},V^B_{t_i}} = \emptyset \vee \mathcal{F}^{i,i+1}_{V_{t_{i+1},t_i},V^T_{t_i}} = \{[m_{c_i}, h^*_{d_i}], [\overline{m_{c_i}}, \overline{h^*_{d_i}}]\}$;
- $\mathcal{F}^{i,i+1}_{V^T_{t_{i+1}},V_{t_{i+1},t_i}} = \emptyset \vee \mathcal{F}^{i,i+1}_{V^T_{t_{i+1}},V_{t_{i+1},t_i}} = \{[m_{c_{i+1}}, h^*_{d_{i+1}}], [\overline{m_{c_{i+1}}}, \overline{h^*_{d_{i+1}}}]\}$;
- $v_{d_i} \in V^B_{\pi_i}$, $v_{c_i}, v_{d_{i+1}} \in V_{\pi_{i+1},\pi_i}$ y $v_{c_{i+1}} \in V^T_{\pi_{i+1}}$ (v. Figura 4.9).

Dado que la instancia 2SAT $\mathcal{F}^{i,E}_{V_{\pi_{i+1},\pi_i}}$ (v. (4.8)) no contiene literales p-eliminables, es satisfacible. Entonces también $\mathcal{F}^{i,i+1}_{V_{\pi_{i+1},\pi_i}}$ es satisfacible. Por lo tanto existe una atribución de verdad sobre las variables de esta fórmula proposicional, $var(\mathcal{F}^{i,i+1}_{V_{\pi_{i+1},\pi_i}})$, que es un modelo para esta fórmula. Ampliándola con los literales de los vértices del conjunto

V_{π_{i+1},π_i} atribuidos por las atribuciones $\tau^{i,i+1}$ y τ^i, se obtiene una atribución de verdad τ_{π_{i+1},π_i} satisfaciendo la fórmula $\mathcal{F}^E_{V_{\pi_{i+1},\pi_i}}$. Como $\tau^{0,i} \subset \tau^i$ y $\tau^{0,i+1} \subset \tau^{i,i+1}$, τ_{π_{i+1},π_i} contiene las atribuciones $\tau^{0,i}$ y $\tau^{0,i+1}$.

Para comprobar que esta atribución puede ser independiente de las atribuciones $\tau_{\pi_{i+2},\pi_{i+1}}$ y $\tau_{\pi_i,\pi_{i-1}}$ considérese el resultado siguiente.

Lema 4.10. *Sean un vértice v_{π_i}, la atribución de verdad $\tau^{0,i}$, la propagación unitaria $(\mathcal{F}^{i,E}, \tau^i) = PropUnit(\mathcal{F}^E \cup \mathcal{U}(\tau^{0,i}))$ y la fórmula $\mathcal{F}^i_{V^T_{\pi_i},V^B_{\pi_i}} = \{[m_{c_i}, h^*_{d_i}], [\overline{m_{c_i}}, \overline{h^*_{d_i}}]\}$ (v. condiciones (1) – (9), pg. 136).*

Entonces, para cualquier vértice $v_k \in V^T_{\pi_i}$ adyacente a v_{c_i}, la fórmula elemental $\mathcal{F}^i_{k,c_i}$ no contiene cláusulas con literales sobre las variable m_{c_i} y h_k simultáneamente (v. Figura 4.9).

Demostración. Para la demostración del presente resultado consideremos cada uno de los dos siguientes casos, $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$ y $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$.

Supongamos que $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$. Por reducción al absurdo, consideremos que existe una fórmula elemental $\mathcal{F}_{k,c_i}$ de modo que la correspondiente $\mathcal{F}^1_{k,c_i}$ contenga una cláusula con los literales $m^*_{c_i}$ y h^*_k. $\{v_k, v_{c_i}\} \in A$ y, por lo tanto, una arista de propagación (más concretamente una fórmula III, IV o IX a XII). Como la variable h_k no está atribuida y $v_k < v_{t_i} = v_{\pi_i}$, el vértice v_k no es adyacente a v_{t_i} ($v_k \notin N(V_{t_i})$). Por hipótesis, $v_{c_i} \in V^T_{\pi_i}$, $v_{d_i} \in V^B_{\pi_i}$ y $\mathcal{F}^i_{V^T_{\pi_i},V^B_{\pi_i}}$ contiene cláusulas. Por lo tanto, la atribución $\{\overline{h_{c_i}}\} \subset \tau^i$. Por otro lado, como v_k es adyacente a v_{c_i} pero no a v_{t_i}, $\mathcal{F}_{k,c_i}$ es, necesariamente, una fórmula elemental 3SAT. Pero, en este caso, la fórmula obtenida por resolución de la fórmula elemental $\mathcal{F}_{k,c_i}$ con la atribución $\{\overline{h_{c_i}}\}$ contiene, como mucho, la cláusula $[h_k, m^*_k]$, contrariamente a la hipótesis que habíamos admitido.

Consideremos ahora el segundo de los casos referidos, $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$ y, por reducción al absurdo, supongamos que existe una fórmula elemental $\mathcal{F}_{k,c_i}$ conteniendo una cláusula con un literal sobre la variable m_{c_i} y otro sobre la variable h_k, de modo que $\mathcal{F}^{i,E}_{k,c_i}$ contenga una cláusula con los literales $m^*_{c_i}$ y h^*_k.

De acuerdo a la primera parte de la demostración del Lema 4.7, el literal $\overline{h_{c_i}}$ está atribuido ($\overline{h_{c_i}} \in \tau^i$).

Consideremos en primer lugar el subcaso $v_k > v_{c_i}$. Como $\mathcal{F}_{c_i,k}$ es una fórmula elemental de los tipos IX a XII se tiene que el vértice v_k es adyacente a v_{π_i} ($v_k \in N(v_{\pi_i})$) pudiendo ocurrir que $v_k \in N^*(v_{s_i})$ o que $v_k \in N(v_{s_i})$. En el primer caso, $y^{c_i}_a > y^k_a \geq y^{\pi_i}_a > y^{d_i}_a > y^{c_i}_b > y^k_b$ y, como $y^{s_i}_a > y^k_a \geq y^{\pi_i}_a > y^{s_i}_b > y^{d_i}_a > y^k_b$, tenemos que $\mathcal{F}_{s_i,k}$ es una fórmula elemental VI a XII. Pero, dada la existencia de la cláusula $|C_i| = 3$, esta última es una fórmula elemental 3SAT. Por lo que la variable h_k está atribuida por la atribución τ^i, en contradicción con la hipótesis.

En el caso contrario, $y^{c_i}_a > y^k_a > y^{s_i}_a > y^{\pi_i}_a > y^{s_i}_b > y^{d_i}_a > y^{c_i}_b > y^k_b$, por lo que $\mathcal{F}_{c_i,s_i}$ y $\mathcal{F}_{k,s_i}$ son ambas fórmulas elementales III o IV. Por las

desigualdades anteriores resulta que $y_a^{c_i} > y_a^{s_i} > y_b^{s_i} > y_b^{c_i}$ y que $y_a^k > y_a^{s_i} > y_b^{s_i} > y_b^k$ y, además, debido a la existencia de la cláusula $|C_i| = 3$, ninguna de las fórmulas puede ser ni II ni V. En estas condiciones, como las dos variables h_k y h_{c_i} están atribuidas por la atribución τ^i, $\mathcal{F}_{c_i,k}^{i,E} = \emptyset$, estamos en contradicción con la hipótesis.

En el subcaso contrario, cuando $v_k < v_{c_i}$, $\mathcal{F}_{k,c_i}$ es una fórmula III, IV o 3SAT (IX-XII). Pero, como $\overline{h_{c_i}} \in \tau^i$ y además $\mathcal{F}_{k,c_i}^{i,E}$ contiene una cláusula con los literales $h_k^*, m_{c_i}^*$, esta fórmula elemental sólo puede ser o del tipo III o del tipo IV. Por lo tanto, $y_a^k > y_a^{c_i} \geq y_a^{\pi_i} > y_a^{d_i} > y_b^{c_i} > y_b^k$. Entonces, como los vértices $v_k, v_{s_i}, v_{c_i} \in N(v_{\pi_i})$ y G es un grafo cordal, v_k y v_{s_i} son adyacentes entre sí . Empero, v_k y v_{d_i} (adyacentes entre sí) están en las condiciones del Lema 4.8, por lo que la variable h_k está atribuidas entonces $\mathcal{F}_{c_i,k}^{i,E} = \emptyset$, nuevamente en contradicción con la hipótesis de partida. □

A la vista del resultado anterior, o bien $\mathcal{F}^i_{V^T_{\pi_i},V^B_{\pi_i}} = \emptyset$ o bien, la atribución de verdad de las variables asociadas a los vértices de los conjuntos $V_{\pi_i,\pi_{i+1}}$ y $V^B_{\pi_i}$ son independientes o bien, en caso contrario, $\mathcal{F}^i_{V_{\pi_i,\pi_{i+1}},V^B_{\pi_i}} = \{[m_{c_i}, h^*_{d_i}], [\overline{m_{c_i}}, \overline{h^*_{d_i}}]\}$, por lo que cualquier atribución de verdad para las variables de los vértices del conjunto $V_{\pi_i,\pi_{i+1}}$, excepto posiblemente m_{c_i}, es independiente de la atribución de las variables del conjunto $V^B_{\pi_i}$.

Así, consideremos los conjuntos de vértices V_{π_{i+1},π_i} y V^B_π (v. Figura 4.9), una atribución de verdad $\tau_{\pi_i,\pi_{i+1}}$ satisfaciendo $\mathcal{F}^E_{V_{\pi_i,\pi_{i+1}}}$ y una atribución de verdad $\tau_{V^B_i}$ satisfaciendo $\mathcal{F}^E_{V^B_\pi}$. $\tau_{\pi_i,\pi_{i+1}}$ contiene la atribución $\tau^{0,i}$ así como a todos los literales de $V_{\pi_i,\pi_{i+1}}$ cuya atribución sea consecuencia lógica suya por propagación unitaria en $\mathcal{F}^E$.

Entonces, recursivamente se puede construir una atribución de verdad:

$$\tau = \tau_{V^B_{\pi_1}} \cup \tau_{V_{\pi_1,\pi_2}} \cup \cdots \cup \tau_{V_{\pi_{t-1},\pi_t}} \cup \tau_{V^T_{\pi_t}}$$

satisfaciendo $\mathcal{F}^E$.

Dado que $\mathcal{F}^E$ y $\mathcal{F}^E$ son lógicamente equivalentes, $\mathcal{F}$ es satisfacible. Lo cual es un absurdo, pues habíamos considerado que la fórmula proposicional $\mathcal{F}$ no era satisfacible.

Resultando la contradicción de términos en la suposición de que la fórmula lógicamente equivalente sin literales 2p-eliminables $\mathcal{F}^E = EliminaLiterales(\mathcal{F}, 2)$ no contenga a la cláusula nula. Con esto se ha probado el siguiente resultado:

Teorema 4.11. *Se $\mathcal{F}$ es una instancia de* EOSC-SAT*, $\mathcal{F}$ es satisfacible si y sólo si $\mathcal{F}^E =$* ELIMINALITERALES$(\mathcal{F}, 2)$ *no contiene a la cláusula nula.*

Demostración. Como hemos visto, si $\mathcal{F}$ no es satisfacible, entonces, independientemente de la ordenación de sus cláusulas y literales, $\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F}, 2)$ contiene la cláusula nula. □

Y, considerando la definición de la clase de satisfacibilidad polinomial 2PURL, también se ha demostrado que

Teorema 4.12. EOSC-SAT *es una subclase de 2*PURL*.*

Demostración. El resultado es una consecuencia del Teorema 4.11, observando que aquel resultado no depende de la ordenación de las cláusulas y literales de cada instancia. □

Por lo tanto, el problema EOSC puede ser resuelto en tiempo polinomial, pues cada una de sus instancias puede reducirse a una instancia de EOSC-SAT.

Corolario 4.13. *El problema de* EMPAREJAMIENTO ORTOGONAL SIMPLE EN CILINDRO *(*EOSC*) puede ser resuelto en tiempo polinomial.*

Demostración. Hemos visto que una instancia del problema EOSC se puede reducir a una instancia de EOSC-SAT en tiempo polinomial y, por el Teorema 4.12, que EOSC-SAT es una subclase de 2PURL. Por lo tanto, para una instancia del problema EOSC la pregunta sobre la existencia o no de un emparejamiento ortogonal simple (sin cruces) tiene repuesta en tiempo polinomial. □

4.5. Algoritmo para EOSC

Desde un punto de vista practico, la solución de cada una de las instancias W del problema EOSC, se obtiene a partir de un procedimiento muy simple, que proviene de la metodología desarrollada a lo largo del presente capítulo. Cada instancia del problema geométrico se reduce a una instancia del problema de satisfacibilidad, $\mathcal{F}$, resoluble en tiempo polinomial a partir del algoritmo $\text{ELIMINALITERALES}(\mathcal{F}, 2)$.

En concreto, dada una instancia W del problema EOSC, a cada dos pares de puntos $w_i, w_j \in W$ les corresponde una fórmula elemental de las presentadas en el Cuadro 4.1. Cada una de estas fórmulas elementales es identificable a partir de la posición relativa de los puntos de los dos pares, codificada en dos secuencias de cuatro enteros, la y-secuencia representativa de la ordenación de las ordenadas y la x-secuencia representativa de la ordenación de las abscisas. En el Cuadro 4.2 se presenta

la correspondencia entre cada una de estas secuencias (representativas de todas las posiciones relativas posibles entre dos pares de puntos de una instancia del problema EOSC) y la correspondiente fórmula elemental. La fórmula proposicional $\mathcal{F}$ correspondiente a la instancia referida, W, resulta de la conjunción de todas las fórmulas elementales obtenidas.

Su resolución se obtiene identificando y excluyendo los literales 2p-eliminables, mediante el algoritmo ELIMINALITERALES$(\mathcal{F}, 2)$. De acuerdo al Teorema 4.11, la fórmula proposicional sin literales 2p-eliminables obtenida, $\mathcal{F}^E =$ ELIMINALITERALES$(\mathcal{F}, 2)$, es satisfacible si y sólo si no contiene la cláusula nula. Dado que la fórmula proposicional sin literales 2p-eliminables obtenida, $\mathcal{F}^E$, y la inicial, $\mathcal{F}$, son lógicamente equivalentes (v. Teorema 2.2), $\mathcal{F}$ no es satisfacible si y solo $\mathcal{F}^E$ contiene la cláusula nula.

Por lo tanto, una instancia W del problema de EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL CILINDRO tiene respuesta negativa si $\square \in \mathcal{F}^E$ y afirmativa en caso contrario. Sin embargo, en el caso de una instancia con respuesta afirmativa, el algoritmo ELIMINALITERALES$(\mathcal{F}, 2)$ no permite identificar una solución del problema. Para obtener un emparejamiento ortogonal se ha desarrollado el algoritmo RESUELVE-EOSC-SAT a partir de la metodología presentada en la Sección 4.4. Específicamente, considerando la existencia de la secuencia de vértices $v_{\pi_1}, v_{\pi_2}, \cdots, v_{\pi_t}$ referida en la Figura 4.9.

Consideremos el algoritmo RESUELVE-EOSC-SAT$(\mathcal{F})$ (v. Algoritmo 15). En la primera línea de este algoritmo se comienza por determinar una fórmula proposicional sin literales 2p-eliminables, $\mathcal{F}^E =$ ELIMINALITERALES$(\mathcal{F}, 2)$. En el caso de que la fórmula obtenida contenga la cláusula nula, el algoritmo termina con el mensaje *no satisfacible*.

Para una instancia del problema EOSC-SAT satisfacible ($\square \notin \mathcal{F}^E$) el algoritmo RESUELVE-EOSC-SAT funciona en dos fases. En la primera se identifica cada uno de los vértices de la secuencia $v_{\pi_1}, v_{\pi_2}, \cdots, v_{\pi_t}$ y las respectivas atribuciones $\tau^{0,i} = \{\overline{h_{s_i}}, m^*_{s_i}\}$ o $\tau^{0,i} = \{h_{t_i}, m^*_{t_i}\}$, $i = 1, \cdots, t$, según que $C_i \in (\mathcal{F}^E_{s_i,t_i})^T$ o que $C_i \in (\mathcal{F}^E_{s_i,t_i})^B$, respectivamente. Identificado el primer vértice, en cada iteración se identifica el vértice siguiente de la referida secuencia de acuerdo a las propiedades (1) a (9) en la página 136 y (1) a (6) en la página 141. Al terminar el ciclo de iteraciones, en esta primera fase, la atribución τ contiene cada un de los literales de las atribuciones $\tau^{0,i}$, asociadas a los vértices $v_{\pi_1}, \cdots, v_{\pi_t}$.

En estas condiciones, la fórmula proposicional obtenida por propagación unitaria de esta atribución de verdad tras eliminar los literales p-eliminables es una instancia 2SAT satisfacible (línea 25). Esto es consecuencia del hecho de que cada una de las fórmulas proposicionales

Algoritmo 15 RESUELVE-EOSC-SAT($\mathcal{F}$)

Entrada: Instancia de EOSC-SAT $\mathcal{F}$
Salida: Atribución de verdad τ o mensaje *no satisfacible*

$\mathcal{F}^E = \text{ELIMINALITERALES}(\mathcal{F}, 2)$
if $\square \in \mathcal{F}^E$ **then**
  return No satisfacible
end if
$(s,t) = max_j\{(i,j) : C \in \mathcal{F}^E_{i,j}, i < j, |C| = 3\}$
if $(\mathcal{F}^E_{s,t})^B$ contiene cláusula 3SAT **then**
  $\tau^0 = \{h_t, m^*_t\}$, $m^*_t \in C$
else
  $\tau^0 = \{\overline{h_s}, m^*_s\}$, $m^*_s \in C$
end if
$\tau = \tau^0$
while $\mathcal{F}^E_{V^T_t B}$ contiene cláusulas 3SAT **do**
  $(\mathcal{F}^1, \tau^1) = \text{PROPUNIT}(\mathcal{F}^E \cup U(\tau^0))$
  $(\mathcal{F}^{1,E}) = \text{ELIMINALITERALES}(\mathcal{F}^1)$
  if $\mathcal{F}^{1,E}_{V^T_t}$ contiene alguna cláusula 3SAT **then**
    $(s_1, t_1) = max_{j<t}\{(i,j) : C \in \mathcal{F}^{1,E}_{i,j}, i < j, |C| = 3\}$
    if $(\mathcal{F}^{1,E}_{s_1,t_1})^B$ contiene cláusula 3SAT **then**
      $\tau^0 = \{h_{t_1}, m^*_{t_1}\}$, $m^*_{t_1} \in C$
    else
      $\tau^0 = \{\overline{h_{s_1}}, m^*_{s_1}\}$, $m^*_{s_1} \in C$
    end if
    $(s,t) = (s_1, t_1)$; $\tau = \tau \cup \tau^0$
  end if
end while
$(\mathcal{F}, \tau) = \text{PROPUNIT}(\mathcal{F}^E \cup \mathcal{U}(\tau))$
$\tau^1 = Resuelve2\text{SAT}(\mathcal{F})$
$\tau^1 = \tau \cup \tau^1$
return (τ)

$\mathcal{F}^{i,E}_{V_{\pi_{i+1}}, V_{\pi_i}}$, $i = 1, \cdots, t$, $\mathcal{F}^E_{V^B_{\pi_1}}$ y $\mathcal{F}^{t,E}_{V^T_{\pi_t}}$ son instancias 2SAT satisfacibles y de que las atribuciones de cada una de las fórmulas $\mathcal{F}^E_{V^T_{\pi_i}, V^B_{\pi_i}}$, $i = 1, \cdots, t$ son independientes, según se deduce de los lemas 4.10, 4.6 y 4.8. Por lo que, para concluir, se puede utilizar cualquiera de los algoritmos lineales para resolver instancias 2SAT, obteniéndose una atribución de verdad que constituye un modelo para $\mathcal{F}^E$ y, por tanto, para la fórmula inicial, $\mathcal{F}$.

La complejidad de este algoritmo corresponde a la complejidad del algoritmo ELIMINALITERALES($\mathcal{F}, 2$), que, como vimos, admite una implementación que se ejecuta en tiempo $O(|\mathcal{F}|^4)$.

CAPÍTULO 5

Conclusiones y Problemas Abiertos

En este capítulo, se resumen los resultados de la tesis y se presentan nuevos problemas y líneas de investigación que se abren a partir de este trabajo.

A lo largo de la presente memoria, en el dominio de las líneas de investigación en satisfacibilidad proposicional, hemos presentado una nueva clase de satisfacibilidad polinomial PURL, además de una jerarquía de clases de satisfacibilidad PURL $\subset$ 2PURL $\subset \cdots \subset$ rPURL. Se han presentado también otros resultados: la resolución de dos problemas geométricos en el dominio de la Geometría Computacional, en tiempo polinomial. Estos resultados fueron obtenidos a partir de la reducción de sus instancias a instancias de las clases PURL y 2PURL, resolubles en tiempo polinomial.

El problema de SATISFACIBILIDAD PROPOSICIONAL (SAT) es un problema NP-completo. De hecho, es el primer problema NP-completo conocido [coo71], extremadamente relevante por su interés teórico en el campo de la complejidad computacional y por sus múltiples y diversas aplicaciones prácticas. La profundización del conocimiento de las estructuras del problema de la SATISFACIBILIDAD PROPOSICIONAL y el desarrollo y implementación de algoritmos eficientes para su resolución constituyen dos vertientes interdependientes sobre las cuales se desenvuelve en la actualidad la mayor parte del trabajo de investigación en SAT.

Independientemente de esta importante cuestión acerca de la complejidad de SAT, existen algunas clases de satisfacibilidad proposicional cuyas instancias son resolubles por algoritmos polinomiales. Tomando como referencia a las clases polinomiales *bien conocidas* [FG03], en la Figura 2.1 se presenta un diagrama enumerándolas e indicando sus re-

laciones (de inclusión). Como quedó demostrado en el Capítulo 2, PURL contiene todas las clases polinomiales bien conocidas por lo que, en este sentido, puede ser considerada una superclase de satisfacibilidad polinomial.

La identificación de esta nueva clase de satisfacibilidad proposicional resoluble en tiempo polinomial, PURL, surge en el ámbito del trabajo desarrollado para la resolución de los problemas de emparejamiento ortogonal en el cilindro y en el toro. Posteriormente, las técnicas empleadas fueron aplicadas a la resolución de un problema de etiquetado de un conjunto de puntos alineados. Este encuadramiento permite comprender la caracterización de PURL, definida como una clase de clases polinomiales, a partir del concepto integrador de literal p-eliminable. Recordemos que un literal p-eliminable puede ser reconocido en tiempo lineal, respecto al tamaño de la respectiva fórmula proposicional y que está relacionado con la existencia de ciertos ciclos locales asociados a la no satisfacibilidad.

5.1. PURL

PURL fue definida a partir de los conceptos de literal eliminable y p-eliminable formalizados en el Capítulo 2. Según este concepto, un literal eliminable puede ser excluido de una cláusula obteniéndose una fórmula proposicional más simple, lógicamente equivalente a la inicial. Una instancia $\mathcal{F}$ de $\Omega \subset$ PURL es satisfacible si y sólo si la fórmula proposicional sin literales p-eliminables retornada por el algoritmo ELIMINALITERALES($\mathcal{F}$) no contiene la cláusula nula.

Esta nueva clase de satisfacibilidad proposicional, asociada al algoritmo ELIMINALITERALES($\mathcal{F}$), constituye un abordaje integrador para la resolución, en tiempo polinomial, del problema de satisfacibilidad de las instancias de las clases polinomiales bien conocidas. Todavía, está pendiente el problema de saber previamente si una fórmula $\mathcal{F}$ es o no una instancia de PURL. Este aspecto constituye un impedimento para la aplicación de este algoritmo en la resolución de instancias arbitrarias del problema SAT. Pero no lo es para la resolución de los problemas geométricos, para los cuales el problema del reconocimiento no se plantea, pues se halla resuelto.

El algoritmo ELIMINALITERALES($\mathcal{F}$) es sensible a la entrada, por lo que la fórmula proposicional equivalente sin literales p-eliminables obtenida depende de la ordenación de las cláusulas y literales de $\mathcal{F}$. Para evitar esta situación se desarrolló el algoritmo ELIMINALITERA-

LES+($\mathcal{F}$) que permite detectar y excluir los literales p-eliminables, independientemente de la ordenación de las cláusulas y literales de $\mathcal{F}$. Sin embargo, en este caso, el número de cláusulas puede aumentar respecto al de la entrada. A semejanza de la clase PURL, a partir de este algoritmo se definió la clase que designamos PURL+. Constituye un interesante problema que permanece abierto decidir si esta clase es una ampliación de PURL o si son clases distintas.

El algoritmo ELIMINALITERALES+ presenta características que permiten fácilmente una implementación utilizando computación paralela, pues los literales de cada una de las cláusulas pueden ser comprobados con la fórmula proposicional independientemente de las otras cláusulas. Lo que constituye una interesante línea de desarrollo futuro.

Para una instancia de $\Omega \subset$ PURL, o $\Omega \subset$ PURL+, si la fórmula proposicional sin literales p-eliminables $\mathcal{F}^E$ retornada por el algoritmo ELIMINALITERALES($\mathcal{F}$) o, en su caso, ELIMINALITERALES+($\mathcal{F}$)) contiene la cláusula nula, sabemos que ambas no son satisfacibles. Y, en caso contrario, sabemos que son satisfacibles. Sin embargo, en este caso, no se conoce ningún algoritmo que permita identificar un modelo para $\mathcal{F}$. O para $\mathcal{F}^E$, dado que las dos fórmulas proposicionales son lógicamente equivalentes.

En esta línea de investigación se desarrolla el algoritmo SOLVELIMLIT($\mathcal{F}, r$), utilizado en la resolución de instancias arbitrárias de SAT (v. cuadros 2.1, 2.2 y 2.3). Este algoritmo constituye un primer intento de resolver algunas instancias de rPURL. De hecho, el elevado porcentaje de instancias SAT que logra resolver en las pruebas es un indicador prometedor de la eficacia de los conceptos de literal p-eliminable y rp-eliminable en la resolución del problema SAT y un estímulo para el desarrollo de algoritmos aún más eficientes.

Una interesante línea de investigación pasa por el estudio de la implementación del algoritmo ELIMINALITERALES$_3(\mathcal{F}, r)$ siguiendo un esquema que permita suspender el proceso de detección y exclusión de literales rp-eliminables, guardando cada una de las fórmulas, $(\mathcal{R}(C, l), \tau) =$ ELIMINALITERALES$_3(\mathcal{F} \cup \mathcal{U}(\overline{Flip(C, l)}), r - 1)$, en memoria y retomándolas siempre que suceda una simplificación en una de las cláusulas de $\mathcal{F}$. Esto permitiría una implementación eficiente del algoritmo SOLVELIMLIT($\mathcal{F}, r$). Recordemos que este algoritmo puede ser utilizado para resolver instancias arbitrarias del problema SAT proporcionando como respuesta una de las tres opciones siguientes: un modelo para $\mathcal{F}$, un mensaje *no satisfacible* o un mensaje *desisto*.

Como hemos demostrado en el presente trabajo, PURL contiene a todas las clases de satisfacibilidad polinomiales bien conocidas. Sin embargo, permanece pendiente de estudiar la relación entre PURL y otras clases polinomiales de satisfacilidad, por ejemplo las citadas en la Sección 1.7.1: QUAD [dal96], CoE [WM00b] y la clase que Benhamou [ben04]

designó como $\mathcal{S}$. El estudio de estas relaciones constituye una línea de desarrollo a establecer en el futuro.

5.2. UC-4P

Es sobradamente conocido que la Geometría Computacional posee un vasto campo de aplicaciones en las áreas de los sistemas de información geográfica (SIG), diseño asistido por computador (CAD/CAM), computación gráfica, robótica, electrónica y computación entre muchas otras. En estos dominios, el etiquetado de mapas, el trazado de circuitos integrados o la creación de diagramas de flujos y organigramas constituyen ejemplos de problemas cuyas instancias se pueden reducir al problema de la SATISFACIBILIDAD PROPOSICIONAL. Por lo que, en estos casos, su resolución puede ser conseguida a partir de la resolución del problema de la SATISFACIBILIDAD PROPOSICIONAL de las respectivas fórmulas proposicionales.

La codificación de algunos tipos de problemas, geométricos o no, utilizando variables booleanas y, a partir de esta codificación, reducir instancias de estos problemas a instancias SAT es una estrategia conocida y frecuentemente utilizada. En el Capítulo 3 consideramos el problema de etiquetado UC-4P, en el cual cada entrada es un conjunto de puntos sobre una recta de pendiente unitaria y un conjunto de etiquetas cuadradas, de modo que alguno de sus vértices coincida con el punto al que se refiere la etiqueta. En ese mismo capítulo mostramos que este problema es resoluble en tiempo polinomial. A este efecto se probó que la familia UC-4P-SAT, de las fórmulas proposicionales que corresponden a alguna instancia de UC-4P, es una subclase de PURL.

Con la resolución del problema de etiquetado de puntos alineados UC-4P se abre una nueva línea de investigación para la resolución de problemas similares a partir de la reducción de las instancias de este problema geométrico a instancias del problema SAT. Como primer ejemplo, podemos considerar el problema UR-4P que constituye una variante del problema UC-4P pero considerando etiquetas rectangulares. Por cada par de puntos de una instancia de este problema, $P_i < P_j$, ordenados de izquierda a derecha según el valor de la abscisa, existen 16 posibles posiciones topológicamente distintas, que conducen a otras tantas fórmulas proposicionales elementales. La Figura 5.1 presenta estas 16 posiciones topológicamente distintas y el Cuadro 5.1, las correspon-

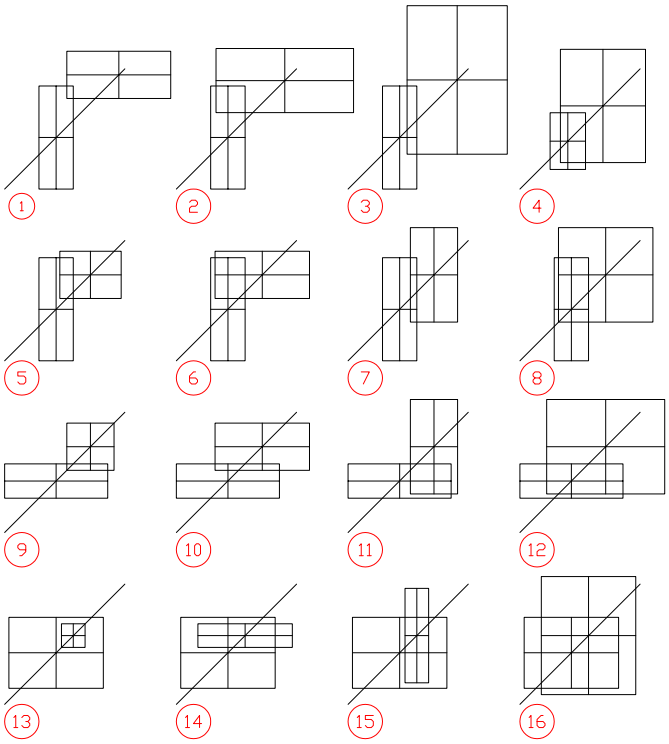


Figura 5.1: Posiciones relativas (fundamentales) de los rectángulos R_i y R_j asociados a las etiquetas de los puntos P_i y P_j.

dientes fórmulas elementales $\mathcal{F}_{i,j}$.

La formulación presentada (UR-4P) es equivalente a la propuesta por Reyes [rey02] (OR-4P), dado que, mediante un cambio de escala apropiado, cualquier instancia del problema OR-4P puede ser transformada en una instancia de UR-4P y viceversa.

UR-4P

Entrada: Un conjunto de puntos, $\mathcal{P} = \{P_1, \cdots, P_n\}$, sobre una recta oblicua de pendiente unitaria y un conjunto de etiquetas rectangulares, $\mathcal{L} = \{L_1, \cdots, L_n\}$.

Pergunta: ¿Puede colocarse el rectángulo L_i con algún de sus vértices coincidente con el punto P_i sin que dos rectángulos se intersequen?

Cuadro 5.1: Fórmulas proposicionales elementales para el problema UR-4P

N.	$\mathcal{F}_{i,j}$	N.	$\mathcal{F}_{i,j}$
1	$\{[l_i, b_i, \overline{l_j}, \overline{b_j}]\}$	9	$\{[l_i, b_i, \overline{b_j}]\}$
2	$\{[b_i, \overline{l_j}, \overline{b_j}]\}$	10	$\{[b_i, \overline{l_j}, \overline{b_j}], [l_i, b_i, \overline{b_j}]\}$
3	$\{[l_i, \overline{l_j}, \overline{b_j}]\}$	11	$\{[l_i, \overline{b_j}]\}$
4	$\{[\overline{l_j}, \overline{b_j}]\}$	12	$\{[l_i, \overline{b_j}], [\overline{l_j}, \overline{b_j}]\}$
5	$\{[l_i, b_i, \overline{l_j}]\}$	13	$\{[l_i, b_i]\}$
6	$\{[b_i, \overline{l_j}]\}$	14	$\{[b_i, \overline{l_j}], [l_i, b_i]\}$
7	$\{[l_i, \overline{l_j}, \overline{b_j}], [l_i, b_i, \overline{l_j}]\}$	15	$\{[l_i, \overline{b_j}], [l_i, b_i]\}$
8	$\{[\overline{l_j}, \overline{b_j}], [b_i, \overline{l_j}]\}$	16	$\{[l_i, b_i], [l_i, \overline{b_j}], [b_i, \overline{l_j}], [\overline{l_j}, \overline{b_j}]\}$

Imponiendo a las condiciones del problema UR-4P la condición adicional de que la altura del rectángulo no puede exceder a su ancho (rectángulo horizontal), se obtiene el subproblema, UR_{Hor}-4P-SAT, cuyas fórmulas proposicionales son conjunciones de las fórmulas elementales 1, 2, 4, 9, 10, 12, 13, 14 y 16 del Cuadro 5.1.

Estos problemas presentados constituyen generalizaciones naturales del problema de etiquetado resuelto en este trabajo, UC-4P y constituyen problemas abiertos en el sentido que explicaremos a continuación. Sabemos ahora que la clase de satisfacibilidad UC-4P es una subclase de PURL y por tanto resoluble en tiempo polinomial. Queda por establecer si las clases UR_{Hor}-4P-SAT y UR-4P-SAT son o no subclases de alguna de las clases de la jerarquía presentada, PURL, 2PURL, $\cdots$, rPURL, $\cdots$. Se conjetura que la primera sea subclase de PURL y la segunda lo sea de 2PURL.

El problema de decisión de etiquetado de puntos en el plano con etiquetas fijas es un problema NP-completo [FW91]. Por lo tanto, no se dispone de ningún algoritmo polinomial que lo resuelva.

Una interesante línea de investigación es la aplicación del concepto de literal p-eliminable (o rp-eliminable) al desarrollo de un algoritmo que sea eficiente en un porcentaje elevado de instancias con interés practico. Aún sería más interesante si el algoritmo estuviese orientado a la implementación de aplicaciones en Sistemas de Información Geográfica.

5.3. EOSC

La investigación sobre inmersiones ortogonales de grafos ha adquirido en los últimos años una gran importancia motivada fundamentalmente por la aplicación al diseño de circuitos integrados. Un buen número de trabajos aborda el problema en el plano. Sin embargo, la producción de resultados de inmersiones ortogonales en superficies ha experimentado un enorme crecimiento desde la introducción de las superficies ortogonales estándar (Cobos [cob95] y Garrido [gar97]).

Hemos presentado en el Capítulo 4 el estudio y la resolución del problema de emparejamiento ortogonal en el cilindro, EOSC. Este problema se encuentra en línea con la resolución del problema plano, en 1986, por Raghavan, Cohoon y Sahni [RCS86] y con el trabajo desarrollado por Portillo [por02] y por Garrido, Márquez, Morgana y Portillo [GMM02] sobre el problema de EMPAREJAMIENTO ORTOGONAL SIMPLE en superficies arbitrarias (EOS). El primero de estos es resoluble en tiempo polinomial y el segundo problema es NP-completo. Era un problema abierto en estos trabajos, el estudio del problema en superficies particulares como el cilindro y el toro (EOSC y EOST, respectivamente).

Como referimos anteriormente, el problema EOSC es resoluble en tiempo polinomial, como consecuencia del hecho de que la clase de satisfacibilidad asociada EOSC-SAT es subclase de 2PURL. Permanece como problema abierto averiguar si EOSC-SAT es subclase de PURL o no.

La resolución del problema EOSC a partir de la reducción de sus instancias a la clase de satisfacibilidad EOSC-SAT abre una nueva línea de investigación para la resolución del problema EMPAREJAMIENTO ORTOGONAL EN EL TORO, utilizando este mismo abordaje. Sabemos que la clase EOST-SAT, constituida por fórmulas proposicionales correspondientes a alguna instancia del problema EOST, no es una subclase de PURL. Basta considerar la fórmula proposicional $\mathcal{F}$ en el Anexo A.1 correspondiente a una instancia del problema EOST, obtenida a partir de las fórmulas elementales del Cuadro 1.5. Esta fórmula proposicional no contiene literales p-eliminables. Sin embargo, la fórmula proposicional lógicamente equivalente obtenida por el algoritmo EliminaLiterales$(\mathcal{F}, 2)$ contiene la cláusula nula. Es decir, $\mathcal{F}^E$ y $\mathcal{F}$ no son satisfacibles. Es decir, $\mathcal{F}$ no contiene literales p-eliminables y nos es satisfacible, por tanto no está en PURL. Pero el algoritmo ELIMINALITERALES$(\mathcal{F}, 2)$ la resuelve, por lo que $\mathcal{F}$ está en 2PURL

Considerando la jerarquía PURL, 2PURL, $\cdots$, rPURL, se mantiene como problema abierto saber si existe algún entero r para el cual rPURL contenga a EOST-SAT como subclase. Se conjetura que esta afirmación es verdadera para $r = 3$.

Las fórmulas proposicionales elementales para el problema de EM-

PAREJAMIENTO ORTOGONAL SIMPLES EN EL TORO presentadas en el Cuadro 1.5 en forma normal conjuntiva, pueden ser escritas como disyunciones de equivalencias. Más concretamente, cualquier fórmula elemental es una conjunción de cláusulas binarias de la forma, $[u, v]$, $[u, v \equiv w]$ y $[u \equiv v, w \equiv x]$ (v. Cuadro 5.2).

Cuadro 5.2: Fórmulas proposicionales elementales en el problema EOST ($i < j$).

	$\mathcal{F}_{i,j}$		$\mathcal{F}_{i,j}$
1	$\{[p_i, m_j], [\overline{m_i}, \overline{p_j}]\}$	**2**	$\{[p_i, \overline{m_j}], [m_i, \overline{p_j}]\}$
3	$\{[p_i, \overline{m_j}], [\overline{m_i}, \overline{p_j}]\}$	**4**	$\{[\overline{p_i}, m_j], [\overline{m_i}, \overline{p_j}]\}$
5	$\{[\overline{p_i}, \overline{m_j}], [m_i, \overline{p_j}]\}$	**6**	$\{[\overline{p_i}, \overline{m_j}], [\overline{m_i}, \overline{p_j}]\}$
7	$\{[m_i, h_i \equiv \overline{p_j}], [\overline{m_j}, p_i \equiv h_j,]\}$	**8**	$\{[\overline{m_i}, h_i \equiv \overline{p_j}], [\overline{m_j}, p_i \equiv h_j,]\}$
9	$\{[\overline{m_i}, h_i \equiv \overline{p_j}], [m_j, p_i \equiv h_j,]\}$	**10**	$\{[p_i, h_i \equiv \overline{m_j}], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
11	$\{[p_i, h_i \equiv \overline{m_j}], [\overline{p_j}, m_i \equiv h_j,]\}$	**12**	$\{[p_i, h_i \equiv m_j], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
13	$\{[p_i, h_i \equiv m_j], [\overline{p_j}, m_i \equiv h_j,]\}$	**14**	$\{[\overline{p_i}, h_i \equiv \overline{m_j}], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
15	$\{[\overline{p_i}, h_i \equiv \overline{m_j}], [\overline{p_j}, m_i \equiv h_j,]\}$	**16**	$\{[\overline{p_i}, h_i \equiv m_j], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
17	$\{[\overline{p_i}, h_i \equiv m_j], [\overline{p_j}, m_i \equiv h_j,]\}$		
18	$\{[h_i \equiv \overline{p_j}, m_i \equiv h_j], [h_i \equiv m_j, p_i \equiv h_j,]\}$	**19**	$\{[h_i \equiv \overline{p_j}, m_i \equiv h_j], [h_i \equiv \overline{m_j}, p_i \equiv h_j,]\}$
20	$\{[h_i \equiv \overline{p_j}, m_i \equiv \overline{h_j}], [h_i \equiv m_j, p_i \equiv h_j,]\}$	**21**	$\{[h_i \equiv \overline{p_j}, m_i \equiv \overline{h_j}], [h_i \equiv \overline{m_j}, p_i \equiv h_j,]\}$

Denotemos 2EQUIV-SAT a la clase constituida por las fórmulas proposicionales (arbitrarias) formadas por cláusulas binarias de esta forma. El problema de la satisfacibilidad de las instancias de esta clase es NP-completo. A este efecto, consideremos la transformación f de las instancias del problema de satisfacibilidad NOT-ALL-EQUAL-3SAT en instancias del problema 2EQUIV-SAT, definida a partir de la siguiente correspondencia entre cláusulas, $[u, v, w] \mapsto [u \equiv \overline{v}, u \equiv \overline{w}]$.

NOT-ALL-EQUAL-3SAT
Entrada: Una CNF-fórmula proposicional $\mathcal{F} = \{C_1, C_2, \ldots, C_m\}$ con $|C_i| = 3, i = 1, \ldots, m$.
Pregunta:¿Existe una atribución de verdad de modo que cada cláusula contenga al menos un literal verdadero y un literal falso?

NOT-ALL-EQUAL-3SAT es un problema NP-completo [sch78] y la transformación f, antes definida, polinomial. De acuerdo con a la tabla de verdad presentada en el Cuadro 5.3, podemos concluir que determinar la satisfacibilidad de la instancia $f(\mathcal{F})$ de 2EQUIV-SAT equivale a determinar la satisfacibilidad de la instancia $\mathcal{F}$ del problema NOT-ALL-EQUAL-3SAT. Dado que este es un problema NP-completo, entonces

Cuadro 5.3: Valores de verdad para una cláusula $[u, v, w]$ del problema NOT-ALL-EQUAL-3SAT y para la cláusula $[u \equiv \overline{v}, u \equiv \overline{w}]$

u	v	w	$[u, v, w]$	$[u \equiv \overline{v}, u \equiv \overline{w}]$
1	1	1	0	0
1	1	0	1	1
1	0	1	1	1
1	0	0	1	1
0	1	1	1	1
0	1	0	1	1
0	0	1	1	1
0	0	0	0	0

2EQUIV-SAT es también un problema NP-completo.

Si consideramos que el problema de emparejamiento ortogonal en el cilindro es un caso particular del problema EOST, y que la clase 2EQUIV-SAT es una extensión de la clase EOST-SAT, el resultado anterior permite encuadrar la clase de satisfacibilidad EOST-SAT. Por lo tanto esta clase está entre la clase polinomial, EOSC-SAT y la clase NP-completa 2EQUIV-SAT. Queda, sin embargo, establecer si el problema EOST es o no resoluble en tiempo polinomial.

5.4. Nota final

Han sido varios los resultados presentados a lo largo de la presente memoria. Entre ellos, destacamos por su importancia la presentación de una nueva clase de satisfacibilidad polinomial, PURL, así como de la jerarquía PURL$\subset$ 2PURL $\subset \cdots \subset$ kPURL. Para cualquier k finito, la clase kPURL es resoluble en tiempo polinomial.

PURL contiene todas las clases polinomiales de satisfacibilidad BIEN CONOCIDAS y, en este sentido, constituye una superclase de satisfacibilidad polinomial.

El interés práctico de PURL y de la jerarquía de clases definida ha quedado demostrado mediante sus aplicaciones a la resolución del problema de etiquetado UC-4P y del problema de emparejamiento EOSC, dos problemas geométricos que estaban abiertos hasta la utilización de las técnicas aquí presentadas.

Llegados a esta altura, sin embargo tenemos la percepción de que, a pesar de los desarrollos realizados, queda mucho por hacer. Nuevas

cuestiones y problemas por estudiar (y resolver) abren nuevas líneas de investigación que, sin duda, proveerán futuros desarrollos en este campo.

CAPÍTULO 6

Conclusões e Problemas Abertos

No presente capítulo resumem-se alguns dos principais resultados desta dissertação e apresentam-se novos problemas e linhas de investigação que se abrem a partir do presente trabalho.

Ao longo da presente dissertação, no domínio das linhas de investigação em satisfação proposicional apresentámos um nova classe de satisfação polinomial PURL bem como uma hierarquia de classes de satisfação PURL $\subset$ 2PURL $\subset \cdots \subset$ rPURL. Apresentámos também a resolução de dois problemas geométricos no domínio da Geometria Computacional, em tempo polinomial. Estes resultados foram obtidos a partir da redução das suas instâncias a instâncias das classes PURL e 2PURL, resolúveis em tempo polinomial.

O problema da SATISFAÇÃO PROPOSICIONAL (SAT) é um problema NP-completo. De facto, SAT foi o primeiro problema NP-completo conhecido [coo71], relevante pelo seu interesse teórico no domínio da complexidade computacional e pelas suas múltiplas e diversas aplicações práticas. O aprofundamento do conhecimento das estruturas do problema da SATISFAÇÃO PROPOSICIONAL e o desenvolvimento e implementação de algoritmos eficientes para a sua resolução constituem duas vertentes interdependentes sobre as quais se desenvolve, actualmente, a maior parte do trabalho de investigação em SAT.

Independentemente desta importante questão, acerca da complexidade de SAT, existem algumas classes de satisfação proposicional cujas instâncias são resolúveis por algoritmos polinomiais. Tomando como referência as classes polinomiais *bem conhecidas* [FG03], na Figura 2.1 apresenta-se um diagrama enumerando-as e indicando as suas relações (de inclusão). Como ficou demonstrado no Capítulo 2, PURL contém todas as classes polinomiais bem conhecidas pelo que, neste sentido, pode

ser considerada uma superclasse de satisfação polinomial.

A identificação desta nova classe de satisfação proposicional resolúvel em tempo polinomial, PURL, surge no âmbito do trabalho desenvolvido para a resolução dos problemas de emparelhamento ortogonal no cilindro e no toro. Posteriormente, as técnicas desenvolvidas foram aplicadas na resolução de um problema de etiquetado de um conjunto de pontos alinhados. Este enquadramento permite compreender a caracterização de PURL, definida como uma classe de classes polinomiais, a partir do conceito integrador de literal p-eliminável. Recordemos que um literal p-eliminável pode ser reconhecido em tempo linear relativamente ao tamanho de respectiva fórmula proposicional e que está relacionado com a existência de certos ciclos locais associados à não satisfação.

6.1. PURL

PURL foi definida a partir dos novos conceitos de literal eliminável e p-eliminável formalizados no Capítulo 2. Segundo este conceito, um literal eliminável pode ser excluído de uma cláusula obtendo-se uma fórmula proposicional mais simples, logicamente equivalente à inicial. Uma instância $\mathcal{F}$ de $\Omega \subset$ PURL é satisfazível se e só se a formula proposicional sem literais p-elimináveis devolvida pelo algoritmo ELIMINALITERALES($\mathcal{F}$) não contém a cláusula nula.

Esta nova classe de satisfação proposicional, associada ao algoritmo ELIMINALITERALES($\mathcal{F}$), constitui uma abordagem integradora para a resolução, em tempo polinomial, do problema de satisfação das instâncias das classes polinomiais bem conhecidas. Todavia, está dependente de previamente se saber se $\mathcal{F}$ é ou não uma instância de PURL. Este aspecto constitui uma limitação para a aplicação deste algoritmo na resolução de instâncias arbitrárias do problema SAT. Mas não para a resolução de problemas geométricos, para os quais o problema do reconhecimento geralmente não se coloca (porque está resolvido à partida).

O algoritmo ELIMINALITERALES($\mathcal{F}$) é sensível à entrada, pelo que a fórmula proposicional equivalente sem literais p-elimináveis obtida depende da ordenação das cláusulas e literais de $\mathcal{F}$. Para evitar esta situação desenvolveu-se o algoritmo ELIMINALITERALES$+(\mathcal{F})$ que permite detectar e excluir os literais p-elimináveis, independentemente da ordenação das cláusulas e literais de $\mathcal{F}$. Todavia, neste caso, o número de cláusulas poderá aumentar relativamente ao número de cláusulas da fórmula inicial (entrada). À semelhança da classe PURL, a partir deste algoritmo definiu-se a classe que designámos PURL+. Constitui um in-

teressante problema decidir se esta classe é uma ampliação de PURL ou se são classes distintas. Problema que, todavia, permanece aberto.

O algoritmo ELIMINALITERALES+ apresenta características que facilmente permitem uma implementação utilizando computação paralela, pois os literais de cada uma das cláusulas podem ser testados independentemente das outras cláusulas. O que constitui uma interessante linha de desenvolvimento futuro.

Para uma instância de $\Omega \subset$ PURL, ou $\Omega \subset$ PURL+, se a fórmula proposicional sem literais p-eliminávels $\mathcal{F}^E$ devolvida pelo algoritmo ELIMINALITERALES($\mathcal{F}$), respectivamente ELIMINALITERALES + ($\mathcal{F}$), contém a cláusula nula, sabemos que ambas não são satisfazíveis. E, no caso contrário, sabemos que são satisfazíveis. Todavia, neste caso, não de conhece nenhum algoritmo que permita identificar um modelo para $\mathcal{F}$. Ou para $\mathcal{F}^E$, dado que as duas fórmulas proposicionais são logicamente equivalentes.

Nesta linha de investigação desenvolveu-se o algoritmo SOLVELIMLIT($\mathcal{F}, r$), utilizado na resolução de instâncias arbitrárias de SAT (v. tabelas 2.1, 2.2 e 2.3). Este algoritmo constitui uma primeira tentativa para a resolução de algumas instâncias de rPURL. A elevada percentagem de instâncias SAT resolvidas constitui um promissor indicador da eficácia do conceito de literal p-eliminável e rp-eliminável na resolução do problema SAT e um estímulo para o desenvolvimento de algoritmos e implementações mais eficientes.

Uma interessante linha de investigação passa pelo estudo da implementação do algoritmo ELIMINALITERALES$_3(\mathcal{F}, r)$ seguindo um esquema que permita suspender o processo de identificação e exclusão de literais rp-eliminávels, guardando cada uma das fórmulas, $(\mathcal{R}(C, l), \tau) =$ ELIMINALITERALES$_3(\mathcal{F} \cup \mathcal{U}(\overline{Flip(C, l)}), r-1)$, em memória e retomando-a sempre que ocorra uma simplificação numa das cláusulas de $\mathcal{F}$. O que permitiria uma implementação eficiente do algoritmo SOLVELIMLIT($\mathcal{F}, r$). Recordemos que este algoritmo pode ser utilizado para resolver instâncias arbitrárias do problema SAT, proporcionando como resposta uma das três opções seguintes: um modelo para $\mathcal{F}$, a mensagem *não satisfazível* ou a mensagem *desisto*.

Como demonstrámos no presente trabalho, PURL contém todas as classes de satisfação polinomial bem conhecidas. Contudo, ficou por estudar a relação entre PURL e outras classes, nomeadamente as citadas na Secção 1.7.1: QUAD [dal96], CoE [WM00b] e a classe que Benhamou [ben04] designou por $\mathcal{S}$. O estudo destas relações constitui uma linha de estudo a desenvolver no futuro.

6.2. UC-4P

É sobejamente conhecido que a Geometria Computacional possui um vasto campo de aplicações nas áreas dos sistemas de informação geográfica (SIG), desenho assistido por computador (CAD/CAM), computação gráfica, robótica, electrónica e computação, entre muitas outras. Nestes domínios, etiquetado de mapas, traçados de circuitos integrados ou criação de diagramas de fluxos e organogramas constituem exemplos de problemas cujas instâncias se podem reduzir ao problema da SATISFAÇÃO PROPOSICIONAL. Pelo que, nestes casos, a sua resolução pode ser conseguida a partir da resolução do problema da SATISFAÇÃO PROPOSICIONAL das respectivas fórmulas proposicionais.

A codificação de alguns tipos de problemas, geométricos ou não, utilizando variáveis booleanas e, a partir dela, reduzir instâncias destes problemas a instâncias SAT é uma estratégia conhecida frequentemente utilizada. No Capítulo 3 considerámos o problema de etiquetado UC-4P, no qual cada entrada é constituída por um conjunto de pontos sobre uma recta de declive unitário e um conjunto de etiquetas quadradas, de modo que algum dos seus vértices coincida com o ponto a que se refere a etiqueta. Neste mesmo capítulo mostrámos que este problema é resolúvel em tempo polinomial. Para o efeito demonstrou-se que a família UC-4P-SAT, constituída pelas fórmulas proposicionais que correspondem a alguma instância de UC-4P, é uma subclasse de PURL.

Com a resolução do problema de etiquetado de pontos alinhados UC-4P abre-se una nova linha de investigação para a resolução do problema similares a partir da redução das instâncias deste problema geométrico a instâncias do problema SAT. Como primeiro exemplo, podemos considerar o problema UR-4P que constitui uma variante do problema UC-4P considerando etiquetas rectangulares. Por cada par de pontos de uma instância deste problema, $P_i < P_j$, ordenados da esquerda para a direita segundo o valor da abcissa, existem 16 possíveis posições topologicamente distintas, que conduzem a outras tantas fórmulas proposicionais elementares. A Figura 6.1 apresenta estas 16 posições topologicamente distintas e a Tabela 6.1, as correspondentes fórmulas elementares $\mathcal{F}_{i,j}$.

A formulação apresentada (UR-4P) é equivalente à proposta por Reyes [rey02] (OR-4P), dado que, para uma mudança de escala conveniente qualquer instância do problema OR-4P pode ser convertida numa instância de UR-4P e vice-versa.

UR-4P
Entrada: Um conjunto de pontos, $\mathcal{P} = \{P_1, \cdots, P_n\}$, sobre uma recta

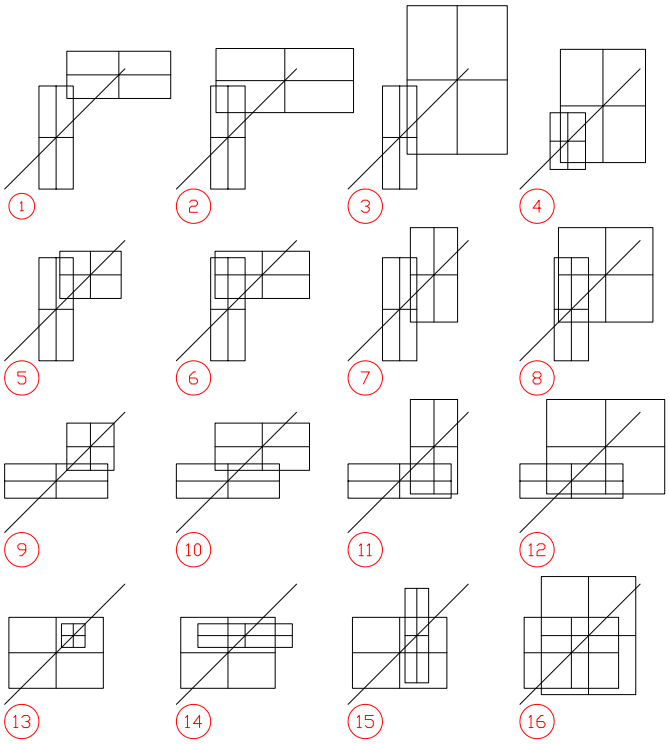


Figura 6.1: Posições relativas (fundamentais) dos rectângulos R_i e R_j associados às etiquetas dos pontos P_i e P_j.

oblíqua de decline unitário e um conjunto de etiquetas rectangulares, $\mathcal{L} = \{L_1, \cdots, L_n\}$.
Pergunta: Pode colocar-se o rectângulo L_i, com algum dos seus vértices coincidente com P_i, sem que dois rectângulos se intersectem?

Impondo às condições do problema UR-4P a condição adicional de que a altura do rectângulo não exceda a sua largura (rectângulo horizontal), obtém-se o (sub)problema, UR_{Hor}-4P-SAT, cujas fórmulas proposicionais são conjunções das formulas elementares 1, 2, 4, 9, 10, 12, 13, 14 e 16 da Tabela 5.1.

Estes problemas apresentados constituem naturais generalizações do problema de etiquetado resolvido no presente trabalho, UC-4P, e cons-

Tabela 6.1: Fórmulas proposicionais elementares para o problema UR-4P

N.	$\mathcal{F}_{i,j}$	N.	$\mathcal{F}_{i,j}$
1	$\{[l_i, b_i, \overline{l_j}, \overline{b_j}]\}$	9	$\{[l_i, b_i, \overline{b_j}]\}$
2	$\{[b_i, \overline{l_j}, \overline{b_j}]\}$	10	$\{[b_i, \overline{l_j}, \overline{b_j}], [l_i, b_i, \overline{b_j}]\}$
3	$\{[l_i, \overline{l_j}, \overline{b_j}]\}$	11	$\{[l_i, \overline{b_j}]\}$
4	$\{[\overline{l_j}, \overline{b_j}]\}$	12	$\{[l_i, \overline{b_j}], [\overline{l_j}, \overline{b_j}]\}$
5	$\{[l_i, b_i, \overline{l_j}]\}$	13	$\{[l_i, b_i]\}$
6	$\{[b_i, \overline{l_j}]\}$	14	$\{[b_i, \overline{l_j}], [l_i, b_i]\}$
7	$\{[l_i, \overline{l_j}, \overline{b_j}], [l_i, b_i, \overline{l_j}]\}$	15	$\{[l_i, \overline{b_j}], [l_i, b_i]\}$
8	$\{[\overline{l_j}, \overline{b_j}], [b_i, \overline{l_j}]\}$	16	$\{[l_i, b_i], [l_i, \overline{b_j}], [b_i, \overline{l_j}], [\overline{l_j}, \overline{b_j}]\}$

tituem problemas abertos no sentido que explicamos a seguir. Sabemos agora que a classe de satisfação UC-4P-SAT é uma subclasse de PURL e, portanto, resolúvel em tempo polinomial. Fica por estabelecer se as subclasses UR_{Hor}-4P-SAT e UR-4P-SAT são ou não subclasses de alguma das classes da hierarquia apresentada, PURL, 2PURL, $\cdots$, rPURL, ... Conjectura-se que a primeira seja subclasse de PURL e a segunda de 2PURL.

O problema de decisão de etiquetado de pontos no plano com etiquetas fixas é um problema NP-completo [FW91]. Pelo que não é conhecido nenhum algoritmo polinomial que o resolva.

Uma interessante linha de investigação é a aplicação do conceito de literal p-eliminável (ou rp-eliminável) para o desenvolvimento de um algoritmo que seja eficiente para uma elevada percentagem de instâncias do problema de etiquetado geral. Que apresentaria especial interesse se for orientado para uma implementação em de Sistemas de Informação Geográfica.

6.3. EOSC

A investigação sobre imersões ortogonais de grafos tem adquirido nos últimos anos uma grande importância motivada fundamentalmente pela aplicação no desenho de circuitos integrados. Um bom número de trabalhos aborda o problema no plano. Contudo, a produção de resultados de imersões ortogonais em superfícies tem crescido grandemente desde a introdução das superfícies ortogonais standard (Cobos e de Gar-

rido [cob95, gar97]).

No Capítulo 4 apresentámos o estudo e a resolução do problema de emparelhamento ortogonal no cilindro, EOSC. Problema na linha de investigação do problema no plano resolvido por Raghavan, Cohoon e Sahni [RCS86] e do trabalho desenvolvido por Portillo [por02] e por Garrido, Márquez, Morgana e Portillo [GMM02] sobre o problema de EMPARELHAMENTO ORTOGONAL SIMPLES em superfícies (EOS). O primeiro resolúvel em tempo polinomial e o segundo NP-completo. Era um problema aberto nestes trabalhos o estudo do problema em superfícies particulares como o cilindro e o toro (EOSC e EOST, respectivamente).

Como referimos anteriormente, o problema EOSC é resolúvel em tempo polinomial, em consequência do facto da classe de satisfação EOSC-SAT constituir uma subclasse de 2PURL. Permanece como problema aberto averiguar se EOSC-SAT é ou não subclasse de PURL.

A resolução do problema EOSC a partir da redução das suas instâncias à classe de satisfação EOSC-SAT abre uma nova linha de investigação para a resolução do problema de EMPARELHAMENTO ORTOGONAL SIMPLES NO TORO, utilizando esta mesma abordagem. Sabemos que a classe EOST-SAT, constituída por formula proposicionais correspondentes a alguma instância do problema EOST, não é uma subclasse de PURL. Basta considerar a formula proposicional $\mathcal{F}$ (A.1), correspondente a uma instância do problema EOST, obtida a partir das fórmulas elementares da Tabela 1.5. Esta fórmula proposicional não contém literais p-eliminàveis. Todavia, a fórmula proposicional logicamente equivalente obtida pelo algoritmo EliminaLiterales$(\mathcal{F}, 2)$ contém a cláusula nula, pelo que $\mathcal{F}^E$ e $\mathcal{F}$ não são satisfazíveis. I. e., $\mathcal{F}$ não contém literais p-eliminàveis e não é satisfazível, portanto não pertence a PURL. Contudo, o algoritmo EliminaLiterales$(\mathcal{F}, 2)$ resolve esta instância pelo que $\mathcal{F}$ pertence a 2PURL.

Considerando a hierarquia PURL, 2PURL, $\cdots$, rPURL, mantém-se como problema aberto saber se existe algum inteiro r para o qual rPURL contenha EOST-SAT como subclasse. Conjectura-se que esta afirmação seja verdadeira para $r = 3$.

As fórmulas proposicionais elementares para o problema de EMPARELHAMENTO ORTOGONAL SIMPLES NO TORO apresentadas na Tabela 1.5 na forma normal conjuntiva, podem ser escritas como disjunções de equivalências. Mais concretamente, qualquer fórmula elementar é uma conjunção de cláusulas binárias da forma, $[u, v]$, $[u, v \equiv w]$ e $[u \equiv v, w \equiv x]$ (v. Tabela 6.2).

Denotemos 2EQUIV-SAT a classe constituída pelas fórmulas proposicionais (arbitrárias) constituídas por cláusulas binárias desta forma. O problema da satisfação das instâncias desta classe é NP-completo. Para o efeito consideremos a transformação f das instâncias do problema de

Tabela 6.2: Fórmulas proposicionais elementares para o problema EOST ($i < j$).

	$\mathcal{F}_{i,j}$		$\mathcal{F}_{i,j}$
1	$\{[p_i, m_j], [\overline{m_i}, \overline{p_j}]\}$	**2**	$\{[p_i, \overline{m_j}], [m_i, \overline{p_j}]\}$
3	$\{[p_i, \overline{m_j}], [\overline{m_i}, \overline{p_j}]\}$	**4**	$\{[\overline{p_i}, m_j], [\overline{m_i}, \overline{p_j}]\}$
5	$\{[\overline{p_i}, \overline{m_j}], [m_i, \overline{p_j}]\}$	**6**	$\{[\overline{p_i}, \overline{m_j}], [\overline{m_i}, \overline{p_j}]\}$
7	$\{[m_i, h_i \equiv \overline{p_j}], [\overline{m_j}, p_i \equiv h_j,]\}$	**8**	$\{[\overline{m_i}, h_i \equiv \overline{p_j}], [\overline{m_j}, p_i \equiv h_j,]\}$
9	$\{[\overline{m_i}, h_i \equiv \overline{p_j}], [m_j, p_i \equiv h_j,]\}$	**10**	$\{[p_i, h_i \equiv \overline{m_j}], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
11	$\{[p_i, h_i \equiv \overline{m_j}], [\overline{p_j}, m_i \equiv h_j,]\}$	**12**	$\{[p_i, h_i \equiv m_j], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
13	$\{[p_i, h_i \equiv m_j], [\overline{p_j}, m_i \equiv h_j,]\}$	**14**	$\{[\overline{p_i}, h_i \equiv \overline{m_j}], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
15	$\{[\overline{p_i}, h_i \equiv \overline{m_j}], [\overline{p_j}, m_i \equiv h_j,]\}$	**16**	$\{[\overline{p_i}, h_i \equiv m_j], [\overline{p_j}, \overline{m_i} \equiv h_j,]\}$
17	$\{[\overline{p_i}, h_i \equiv m_j], [\overline{p_j}, m_i \equiv h_j,]\}$		
18	$\{[h_i \equiv \overline{p_j}, m_i \equiv h_j], [h_i \equiv m_j, p_i \equiv h_j,]\}$	**19**	$\{[h_i \equiv \overline{p_j}, m_i \equiv h_j], [h_i \equiv \overline{m_j}, p_i \equiv h_j,]\}$
20	$\{[h_i \equiv \overline{p_j}, m_i \equiv \overline{h_j}], [h_i \equiv m_j, p_i \equiv h_j,]\}$	**21**	$\{[h_i \equiv \overline{p_j}, m_i \equiv \overline{h_j}], [h_i \equiv \overline{m_j}, p_i \equiv h_j,]\}$

satisfação NOT-ALL-EQUAL-3SAT em instâncias do problema 2EQUIV-SAT, definida a partir da seguinte correspondência entre cláusulas, $[u, v, w] \mapsto [u \equiv \overline{v}, u \equiv \overline{w}]$.

NOT-ALL-EQUAL-3SAT
Entrada: Uma CNF-fórmula proposicional $\mathcal{F} = \{C_1, C_2, \ldots, C_m\}$ com $|C_i| = 3, i = 1, \ldots, m$.
Pergunta: Existe alguma atribuição de verdade que satisfaça $\mathcal{F}$, de modo que cada cláusula contenha pelo menos um literal verdadeiro e um literal falso?

NOT-ALL-EQUAL-3SAT é um problema NP-completo [sch78] e a transformação f, atrás definida, uma transformação polinomial. De acordo com a tabela de verdade apresentada na Tabela 6.3, podemos concluir que determinar a satisfação da instância $f(\mathcal{F})$ de 2EQUIV-SAT equivale a determinar a satisfação da instância $\mathcal{F}$ do problema NOT-ALL-EQUAL-3SAT. Dado que este +e um problema NP-completo, então 2EQUIV-SAT é também um problema NP-completo.

Se considerarmos que o problema de emparelhamento ortogonal no cilindro constitui um caso particular do problema EOST, e que a classe 2EQUIV-SAT constitui uma extensão da classe EOST-SAT, o anterior resultado permite enquadrar a classe de satisfação EOST-SAT. Pelo que esta classe está entre a classe polinomial, EOSC-SAT, e a classe NP-completa 2EQUIV-SAT. Fica contudo por estabelecer se o problema EOST é ou não resolúvel em tempo polinomial.

Tabela 6.3: Valores de verdade para uma cláusula $[u, v, w]$ do problema NOT-ALL-EQUAL-3SAT e para a cláusula $[u \equiv \overline{v}, u \equiv \overline{w}]$.

u	v	w	$[u, v, w]$	$[u \equiv \overline{v}, u \equiv \overline{w}]$
1	1	1	0	0
1	1	0	1	1
1	0	1	1	1
1	0	0	1	1
0	1	1	1	1
0	1	0	1	1
0	0	1	1	1
0	0	0	0	0

6.4. Nota final

São vários os resultados apresentados ao longo da presente dissertação. Entre eles, destacamos pela sua importância a apresentação da nova classe de satisfação polinomial: PURL bem como a apresentação da hierarquia PURL $\subset$ 2PURL $\subset \cdots \subset$ kPURL. Para cada k finito a classe kPURL é resolúvel em tempo polinomial.

PURL, contém todas as classes de satisfação polinomial bem conhecidas, pelo que, neste sentido, constitui uma superclasse de satisfação polinomial.

O interesse pratico de PURL e da hierarquia de classes definida ficou comprovado pelas suas aplicações na resolução do problema de etiquetado UC-4P e do problema de emparelhamento EOSC, dois problemas geométricos que estavam por resolver até à utilização das técnicas aqui apresentadas.

Chegados a esta altura, apesar dos desenvolvimento conseguidos temos a percepção que muito fica por fazer. Novas questões e problemas que ficam por estudar e resolver abrem diversas linhas de investigação que poderão motivar desenvolvimentos futuros neste campo.

APÉNDICE A

Anexos

A.1. Instancia de EOST no satisfacible

$\mathcal{F} =$ {[3, 6, −1, 5], [3, −6, 1, 5], [−3, 6, −1, −5], [−3, −6, 1, −5], [3, 6, −4, −2], [3, −6, −4, 2],
[−3, −6, 4, 2], [−3, 6, 4, −2], [3, 9, 1, 8], [3, −9, −1, 8], [−3, 9, 1, −8], [−3, −9, −1, −8],
[3, 9, −7, −2], [3, −9, −7, 2], [−3, 9, 7, −2], [−3, −9, 7, 2], [3, 12, 1, 11], [3, −12, −1, 11],
[−3, 12, 1, −11], [−3, −12, −1, −11], [3, 12, −10, −2], [3, −12, −10, 2], [−3, 12, 10, −2], [−3, −12, 10, 2],
[3, 15, −1, 14], [3, −15, 1, 14], [−3, 15, −1, −14], [−3, −15, 1, −14], [3, 15, −13, −2], [3, −15, −13, 2],
[−3, −15, 13, 2], [−3, 15, 13, −2], [3, 18, 1, 17], [3, −18, −1, 17], [−3, 18, 1, −17], [−3, −18, −1, −17],
[3, 18, −16, −2], [3, −18, −16, 2], [−3, 18, 16, −2], [−3, −18, 16, 2], [3, 21, −1, 20], [3, −21, 1, 20],
[−3, 21, −1, −20], [−3, −21, 1, −20], [3, 21, −19, −2], [3, −21, −19, 2], [−3, −21, 19, 2], [−3, 21, 19, −2],
[3, 24, 1, 23], [3, −24, −1, 23], [−3, 24, 1, −23], [−3, −24, −1, −23], [3, 24, −22, −2], [3, −24, −22, 2],
[−3, 24, 22, −2], [−3, −24, 22, 2], [3, 27, 1, 26], [3, −27, −1, 26], [−3, 27, 1, −26], [−3, −27, −1, −26],
[3, 27, −25, −2], [3, −27, −25, 2], [−3, 27, 25, −2], [−3, −27, 25, 2], [3, 30, −1, 29], [3, −30, 1, 29],
[−3, 30, −1, −29], [−3, −30, 1, −29], [3, 30, −28, −2], [3, −30, −28, 2], [−3, −30, 28, 2], [−3, 30, 28, −2],
[3, 33, −1, 32], [3, −33, 1, 32], [−3, 33, −1, −32], [−3, −33, 1, −32], [3, 33, −31, −2], [3, −33, −31, 2],
[−3, −33, 31, 2], [−3, 33, 31, −2], [3, 36, 1, 35], [3, −36, −1, 35], [−3, 36, 1, −35], [−3, −36, −1, −35],
[3, 36, −34, −2], [3, −36, −34, 2], [−3, 36, 34, −2], [−3, −36, 34, 2], [6, 9, 4, 8], [6, −9, −4, 8],
[−6, 9, 4, −8], [−6, −9, −4, −8], [6, 9, 7, −5], [6, −9, 7, 5], [−6, 9, −7, −5], [−6, −9, −7, 5],
[6, 12, 4, 11], [6, −12, −4, 11], [−6, 12, 4, −11], [−6, −12, −4, −11], [6, 12, 10, −5], [6, −12, 10, 5],
[−6, 12, −10, −5], [−6, −12, −10, 5], [6, 15, −4, 14], [6, −15, 4, 14], [−6, 15, −4, −14], [−6, −15, 4, −14],
[6, 15, 13, −5], [6, −15, 13, 5], [−6, 15, −13, −5], [−6, −15, −13, 5], [6, 18, 4, 17], [6, −18, −4, 17],
[−6, 18, 4, −17], [−6, −18, −4, −17], [6, 18, 16, −5], [6, −18, 16, 5], [−6, 18, −16, −5], [−6, −18, −16, 5],
[6, 21, −4, 20], [6, −21, 4, 20], [−6, 21, −4, −20], [−6, −21, 4, −20], [6, 21, 19, −5], [6, −21, 19, 5],
[−6, 21, −19, −5], [−6, −21, −19, 5], [6, 24, 4, 23], [6, −24, −4, 23], [−6, 24, 4, −23], [−6, −24, −4, −23],
[6, 24, 22, −5], [6, −24, 22, 5], [−6, 24, −22, −5], [−6, −24, −22, 5], [6, 27, 4, 26], [6, −27, −4, 26],
[−6, 27, 4, −26], [−6, −27, −4, −26], [6, 27, 25, −5], [6, −27, 25, 5], [−6, 27, −25, −5], [−6, −27, −25, 5],
[6, 30, −4, 29], [6, −30, 4, 29], [−6, 30, −4, −29], [−6, −30, 4, −29], [6, 30, 28, −5], [6, −30, 28, 5],
[−6, 30, −28, −5], [−6, −30, −28, 5], [6, 33, −4, 32], [6, −33, 4, 32], [−6, 33, −4, −32], [−6, −33, 4, −32],
[6, 33, 31, −5], [6, −33, 31, 5], [−6, 33, −31, −5], [−6, −33, −31, 5], [6, 36, 4, 35], [6, −36, −4, 35],
[−6, 36, 4, −35], [−6, −36, −4, −35], [6, 36, 34, −5], [6, −36, 34, 5], [−6, 36, −34, −5], [−6, −36, −34, 5],

[9, 12, 7, 11], [9, −12, −7, 11], [−9, 12, 7, −11], [−9, −12, −7, −11], [9, 12, −10, −8], [9, −12, −10, 8],
[−9, 12, 10, −8], [−9, −12, 10, 8], [9, 15, −7, 14], [9, −15, 7, 14], [−9, 15, −7, −14], [−9, −15, 7, −14],
[9, 15, −13, −8], [9, −15, −13, 8], [−9, −15, 13, 8], [−9, 15, 13, −8], [9, 18, 7, 17], [9, −18, −7, 17],
[−9, 18, 7, −17], [−9, −18, −7, −17], [9, 18, −16, −8], [9, −18, −16, 8], [−9, 18, 16, −8], [−9, −18, 16, 8],
[9, 21, −7, 20], [9, −21, 7, 20], [−9, 21, −7, −20], [−9, −21, 7, −20], [9, 21, −19, −8], [9, −21, −19, 8],
[−9, −21, 19, 8], [−9, 21, 19, −8], [9, 24, 7, 23], [9, −24, −7, 23], [−9, 24, 7, −23], [−9, −24, −7, −23],
[9, 24, −22, −8], [9, −24, −22, 8], [−9, 24, 22, −8], [−9, −24, 22, 8], [9, 27, 7, 26], [9, −27, −7, 26],
[−9, 27, 7, −26], [−9, −27, −7, −26], [9, 27, −25, −8], [9, −27, −25, 8], [−9, 27, 25, −8], [−9, −27, 25, 8],
[9, 30, −7, 29], [9, −30, 7, 29], [−9, 30, −7, −29], [−9, −30, 7, −29], [9, 30, −28, −8], [9, −30, −28, 8],
[−9, −30, 28, 8], [−9, 30, 28, −8], [9, 33, −7, 32], [9, −33, 7, 32], [−9, 33, −7, −32], [−9, −33, 7, −32],
[9, 33, −31, −8], [9, −33, −31, 8], [−9, −33, 31, 8], [−9, 33, 31, −8], [9, 36, 7, 35], [9, −36, −7, 35],
[−9, 36, 7, −35], [−9, −36, −7, −35], [9, 36, −34, −8], [9, −36, −34, 8], [−9, 36, 34, −8], [−9, −36, 34, 8],
[12, 15, −10, 14], [12, −15, 10, 14], [−12, 15, −10, −14], [−12, −15, 10, −14], [12, 15, −13, −11], [12, −15, −13, 11],
[−12, −15, 13, 11], [−12, 15, 13, −11], [12, 18, 10, 17], [12, −18, −10, 17], [−12, 18, 10, −17], [−12, −18, −10, −17],
[12, 18, −16, −11], [12, −18, −16, 11], [−12, 18, 16, −11], [−12, −18, 16, 11], [12, 21, −10, 20], [12, −21, 10, 20],
[−12, 21, −10, −20], [−12, −21, 10, −20], [12, 21, −19, −11], [12, −21, −19, 11], [−12, −21, 19, 11], [−12, 21, 19, −11],
[12, 24, 10, 23], [12, −24, −10, 23], [−12, 24, 10, −23], [−12, −24, −10, −23], [12, 24, −22, −11], [12, −24, −22, 11],
[−12, 24, 22, −11], [−12, −24, 22, 11], [12, 27, 10, 26], [12, −27, −10, 26], [−12, 27, 10, −26], [−12, −27, −10, −26],
[12, 27, −25, −11], [12, −27, −25, 11], [−12, 27, 25, −11], [−12, −27, 25, 11], [12, 30, −10, 29], [12, −30, 10, 29],
[−12, 30, −10, −29], [−12, −30, 10, −29], [12, 30, −28, −11], [12, −30, −28, 11], [−12, −30, 28, 11], [−12, 30, 28, −11],
[12, 33, −10, 32], [12, −33, 10, 32], [−12, 33, −10, −32], [−12, −33, 10, −32], [12, 33, −31, −11], [12, −33, −31, 11],
[−12, −33, 31, 11], [−12, 33, 31, −11], [12, 36, 10, 35], [12, −36, −10, 35], [−12, 36, 10, −35], [−12, −36, −10, −35],
[12, 36, −34, −11], [12, −36, −34, 11], [−12, 36, 34, −11], [−12, −36, 34, 11], [15, 18, −13, 17], [15, −18, 13, 17],
[−15, 18, −13, −17], [−15, −18, 13, −17], [15, 18, −16, −14], [15, −18, −16, 14], [−15, −18, 16, 14], [−15, 18, 16, −14],
[15, 21, 13, 20], [15, −21, −13, 20], [−15, 21, 13, −20], [−15, −21, −13, −20], [15, 21, −19, −14], [15, −21, −19, 14],
[−15, 21, 19, −14], [−15, −21, 19, 14], [15, 24, −13, 23], [15, −24, 13, 23], [−15, 24, −13, −23], [−15, −24, 13, −23],
[15, 24, −22, −14], [15, −24, −22, 14], [−15, −24, 22, 14], [−15, 24, 22, −14], [15, 27, −13, 26], [15, −27, 13, 26],
[−15, 27, −13, −26], [−15, −27, 13, −26], [15, 27, −25, −14], [15, −27, −25, 14], [−15, −27, 25, 14], [−15, 27, 25, −14],
[15, 30, 13, 29], [15, −30, −13, 29], [−15, 30, 13, −29], [−15, −30, −13, −29], [15, 30, −28, −14], [15, −30, −28, 14],
[−15, 30, 28, −14], [−15, −30, 28, 14], [15, 33, 13, 32], [15, −33, −13, 32], [−15, 33, 13, −32], [−15, −33, −13, −32],
[15, 33, −31, −14], [15, −33, −31, 14], [−15, 33, 31, −14], [−15, −33, 31, 14], [15, 36, −13, 35], [15, −36, 13, 35],
[−15, 36, −13, −35], [−15, −36, 13, −35], [15, 36, −34, −14], [15, −36, −34, 14], [−15, −36, 34, 14], [−15, 36, 34, −14],
[18, 21, 16, 20], [18, −21, −16, 20], [−18, 21, 16, −20], [−18, −21, −16, −20], [18, 21, 19, −17], [18, −21, 19, 17],
[−18, 21, −19, −17], [−18, −21, −19, 17], [18, 24, −16, 23], [18, −24, 16, 23], [−18, 24, −16, −23], [−18, −24, 16, −23],
[18, 24, 22, −17], [18, −24, 22, 17], [−18, 24, −22, −17], [−18, −24, −22, 17], [18, 27, −16, 26], [18, −27, 16, 26],
[−18, 27, −16, −26], [−18, −27, 16, −26], [18, 27, 25, −17], [18, −27, 25, 17], [−18, 27, −25, −17], [−18, −27, −25, 17],
[18, 30, 16, 29], [18, −30, −16, 29], [−18, 30, 16, −29], [−18, −30, −16, −29], [18, 30, 28, −17], [18, −30, 28, 17],
[−18, 30, −28, −17], [−18, −30, −28, 17], [18, 33, 16, 32], [18, −33, −16, 32], [−18, 33, 16, −32], [−18, −33, −16, −32],
[18, 33, 31, −17], [18, −33, 31, 17], [−18, 33, −31, −17], [−18, −33, −31, 17], [18, 36, −16, 35], [18, −36, 16, 35],
[−18, 36, −16, −35], [−18, −36, 16, −35], [18, 36, 34, −17], [18, −36, 34, 17], [−18, 36, −34, −17], [−18, −36, −34, 17],
[21, 24, −19, 23], [21, −24, 19, 23], [−21, 24, −19, −23], [−21, −24, 19, −23], [21, 24, −22, −20], [21, −24, −22, 20],
[−21, −24, 22, 20], [−21, 24, 22, −20], [21, 27, −19, 26], [21, −27, 19, 26], [−21, 27, −19, −26], [−21, −27, 19, −26],
[21, 27, −25, −20], [21, −27, −25, 20], [−21, −27, 25, 20], [−21, 27, 25, −20], [21, 30, 19, 29], [21, −30, −19, 29],
[−21, 30, 19, −29], [−21, −30, −19, −29], [21, 30, −28, −20], [21, −30, −28, 20], [−21, 30, 28, −20], [−21, −30, 28, 20],
[21, 33, 19, 32], [21, −33, −19, 32], [−21, 33, 19, −32], [−21, −33, −19, −32], [21, 33, −31, −20], [21, −33, −31, 20],
[−21, 33, 31, −20], [−21, −33, 31, 20], [21, 36, −19, 35], [21, −36, 19, 35], [−21, 36, −19, −35], [−21, −36, 19, −35],
[21, 36, −34, −20], [21, −36, −34, 20], [−21, −36, 34, 20], [−21, 36, 34, −20], [24, 27, 22, 26], [24, −27, −22, 26],
[−24, 27, 22, −26], [−24, −27, −22, −26], [24, 27, −25, −23], [24, −27, −25, 23], [−24, 27, 25, −23], [−24, −27, 25, 23],
[24, 30, −22, 29], [24, −30, 22, 29], [−24, 30, −22, −29], [−24, −30, 22, −29], [24, 30, −28, −23], [24, −30, −28, 23],
[−24, −30, 28, 23], [−24, 30, 28, −23], [24, 33, −22, 32], [24, −33, 22, 32], [−24, 33, −22, −32], [−24, −33, 22, −32],
[24, 33, −31, −23], [24, −33, −31, 23], [−24, −33, 31, 23], [−24, 33, 31, −23], [24, 36, 22, 35], [24, −36, −22, 35],
[−24, 36, 22, −35], [−24, −36, −22, −35], [24, 36, −34, −23], [24, −36, −34, 23], [−24, 36, 34, −23], [−24, −36, 34, 23],
[27, 30, −25, 29], [27, −30, 25, 29], [−27, 30, −25, −29], [−27, −30, 25, −29], [27, 30, −28, −26], [27, −30, −28, 26],
[−27, −30, 28, 26], [−27, 30, 28, −26], [27, 33, −25, 32], [27, −33, 25, 32], [−27, 33, −25, −32], [−27, −33, 25, −32],
[27, 33, −31, −26], [27, −33, −31, 26], [−27, −33, 31, 26], [−27, 33, 31, −26], [27, 36, 25, 35], [27, −36, −25, 35],
[−27, 36, 25, −35], [−27, −36, −25, −35], [27, 36, −34, −26], [27, −36, −34, 26], [−27, 36, 34, −26], [−27, −36, 34, 26],
[30, 33, 28, 32], [30, −33, −28, 32], [−30, 33, 28, −32], [−30, −33, −28, −32], [30, 33, −31, −29], [30, −33, −31, 29],
[−30, 33, 31, −29], [−30, −33, 31, 29], [30, 36, −28, 35], [30, −36, 28, 35], [−30, 36, −28, −35], [−30, −36, 28, −35],
[30, 36, −34, −29], [30, −36, −34, 29], [−30, −36, 34, 29], [−30, 36, 34, −29], [33, 36, −31, 35], [33, −36, 31, 35],
[−33, 36, −31, −35], [−33, −36, 31, −35], [33, 36, −34, −32], [33, −36, −34, 32], [−33, −36, 34, 32], [−33, 36, 34, −32]}

Referencias

[AHU74] AHO, A. V., J. E. HOPCROFT, y J. D. ULLMAN. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[asp80] ASPVALL, B. «Recognizing disguised NR(1) instances of the satisfiability problem.» *J. Algorithms*, 1, nº 1, (1980), 97–103.

[APT79] ASPVALL, B., M. F. PLASS, y R. E. TARJAN. «A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas.» *Inf. Process. Lett.*, 8, nº 3, (1979), 121–123.

[bac94] BACHMANN, P. *Die analytische Zahlentheorie*. Teubner, Leipzig, 1894.

[BNT86] BATINI, C., E. NARDELLI, y R. TAMASSIA. «A Layout Algorithm for Data–Flow Diagrams.» *IEEE Trans. on Software Engineering*, SE–12, nº 4, (1986), 538–546.

[BTT84] BATINI, C., M. TALAMO, y R. TAMASSIA. «Computer Aided Layout of Entity–Relationship Diagrams.» *The Journal of Systems and Software*, 4, (1984), 163–173.

[BET99] DI BATTISTA, G., P. EADES, R. TAMASSIA, y I. G. TOLLIS. *Graph Drawing*. Prentice Hall, 1999.

[ari05] BEN-ARI, Mordechai (Moti). «Minesweeper as an NP-complete problem.» *SIGCSE Bull.*, 37, nº 4, (2005), 39–40.

[ben04] BENHAMOU, B. «Satisfiability and matchings in bipartite graphs: relationship and tractability.» *The journal of Spanish Royal Academy of sciences (RACSAM), a special issue on Symbolic Computation and Artificial Intelligence*, 98, nº (1–2), (2004), 55 – 63.

[BKO00] DE BERG, M., M. VAN KREVELD, M. OVERMARS, y O. SCHWARZKOPF. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[bie95] BIEDL, T. C. *Orthogonal Graphs Drawings: algorithms and lower bounds*. Diplomarbeit, Fachbereich Mathematik, Technische Univ. Berlin, 1995. Supervisores: Ralph H. Möring y Gunter M. Ziegler.

[bie96] —. «Optimal orthogonal drawings of triconnected plane graphs.» En *Proc. of SWAT'96* (R. Karlsson, y A. Lingas, eds.). Springer-Verlag, Berlin, 1996, tomo 1097 de *Lectures Notes in Computer Science*, 333–344.

[BHM09] BIERE, A., M. HEULE, H. Van MAAREN, y T. WALSH. *Handbook of Satisfiability*. IOS Press, 2009.

[bol79] BOLLOBAS, B. *Graph Theory*. Graduate texts in Mathematics. Springer Verlag, 1979.

[BJ74] BOOLOS, G. S., y R. C. JEFFREY. *Computability and Logic*. Cambridge University Press, 1974, 2.ª ed$^{\underline{\text{ón}}}$.

[BCH94] BOROS, E., Y. CRAMA, P.L. HAMMER, y M. SAKS. «A Complexity Index for Satisfiability Problems.» *SIAM J. Comput.*, 23, n$^{\underline{\text{o}}}$ 1, (1994), 45–49.

[BHS94] BOROS, E., P.L. HAMMER, y X. SUN. «Recognition of q-Horn formulae in linear time.» *Discrete Appl. Math.*, 55, n$^{\underline{\text{o}}}$ 1, (1994), 1–13.

[BVM84] BRAYTON, Robert King, Alberto L. SANGIOVANNI-VINCENTELLI, Curtis T. MCMULLEN, y Gary D. HACHTEL. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[bry86] BRYANT, Randal E. «Graph-based algorithms for Boolean function manipulation.» *IEEE Trans. Comput.*, 35, n$^{\underline{\text{o}}}$ 8, (1986), 677–691.

[CIR03] CHANDRAN, L. S., L. IBARRA, F. RUSKEY, y J. SAWADA. «Generating and characterizing the perfect elimination orderings of a chordal graph.» *Theor. Comput. Sci.*, 307, n$^{\underline{\text{o}}}$ 2, (2003), 303–317.

[CCH90] CHANDRU, V., C. R. COULLARD, P. L. HAMMER, M. Monta NEZ, y X. SUN. «On renamable Horn and generalized Horn functions.» *Annals of Mathematics and AI*, 1, (1990), 33–48.

[CH91] CHANDRU, V., y J.N.HOOKER. «Extended Horn sets in propositional logic.» *Journal of the Association for Computing Machinery*, 38, n$^{\underline{\text{o}}}$ 1, (1991), 205–221.

[CF86] CHAO, M.T., y J. FRANCO. «Probabilistic Analysis of Two Heuristics for the 3-Satisfiability Problem.» *SIAM J. Comput.*, 15, nº 4, (1986), 1106–1118. URL `citeseer.ist.psu.edu/chao86probabilistic.html`.

[CL86] CHARTRAND, G., y L. LESNIAK. *Graphs and Digraphs*. Wadswoorth & Brooks/Cole, 1986.

[cha96] CHAZELLE, B. «Application challenges to computational geometry.» *Princeton U. Technical Report*, TR-521-96.

[CET99] CHAZELLE, B., y 36 COAUTHORS. «The computational geometry impact task force report.» En *Advances in Discrete and Computational Geometry* (B. Chazelle, J.E. Goodman, y R. Pollack, eds.), $\mathcal{AMS}$ (American Mathematical Society), Providence, RI, EE.UU., 1999, tomo 223. 407–463.

[che03] CHEN, Yu-Shin. *Labeling points on a single line*. Masther thesis, National Taiwan Univ., Dept. Computer Science and Information Engineering, June 2003.

[CLL05] CHEN, Yu-Shin, D. T. LEE, y Chung-Shou LIAO. «Labeling points on a single line.» *International Journal of Computational Geometry and Applications*, 15, nº 3, (2005), 261–277.

[CD86] CHIN, Roland T., y Charles R. DYER. «Model-based recognition in robot vision.» *ACM Comput. Surv.*, 18, nº 1, (1986), 67–108.

[CMS95] CHRISTENSEN, J., J. MARKS, y S. SHIEBER. «An empirical study of algorithms for point-feature label placement.» *ACM Trans. Graph.*, 14, nº 3, (1995), 203–232.

[CM92] CHVÁTAL, V., y B. REED. «Mick Gets Some (the Odds Are on His Side).» En *FOCS*. IEEE, 1992, 620–627.

[cob64] COBHAM, A. «The intrinsic computational difficulty of functions.» En *Proc. 1964 International Congress for Logic Methodology and Philosophy of Science* (Bar-Hilled, ed.). North-Holland, Amsterdam, Nueva York, 1964, 24–30.

[cob95] COBOS GAVALA, F. J. *Sobre visibilidad en espacios n–dimensionales y superficies no planas*. Tesis Doctoral, Universidad de Sevilla, Dpto. de Matemática Aplicada I, 1995.

[CH88] COHOON, James P., y Patrick L. HECK. «A Computational–Geometry–Based Tool for Switchbox Routing.» *IEEE Trans. on Computer–Aided Design*, 7, nº 6, (1988), 684–697.

[CGK94] CONFORTI, M., G. CORNUÉJOLS, A. KAPOOR, K. VUSKOVIĆ, y M. R. RAO. *Balanced matrices*, University of Michigan, 1994 1–33. URL `citeseer.ist.psu.edu/conforti01balanced.html`.

[coo71] COOK, S. A. «The complexity of theorem–proving procedures.» En *Proc. 3rd Ann. ACM Symp. on Theory of Computing* (Association for Computing Machinery, ed.). New York, 1971, 151–158.

[CB94] CRAWFORD, James M., y Andrew B. BAKER. «Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems.» En *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. AAAI Press/MIT Press, Seattle, Washington, USA, 1994, tomo 2, 1092–1097. URL `citeseer.ist.psu.edu/crawford94experimental.html`.

[DE92] DALAL, M., y D.W. ETHERINGTHON. «A hierarchy of tractable satisfiability problems.» *Information Processing Letters*, 44, nº 4, (92), 173–180.

[dal96] DALAL, Mukesh. «An Almost Quadratic Class of Satisfiability Problems.» En *ECAI* (Wolfgang Wahlster, ed.). John Wiley and Sons, Chichester, 1996, 355–359.

[DLL62] DAVIS, M., G. LOGEMANN, y D. LOVELAND. «A Machine Program for Theorem-Proving.» *Communications of the Association for Computing Machinery*, 5, nº 7, (1962), 394–397.

[DP60] DAVIS, M., y H. PUTNAM. «A computing procedure for quantification theory.» *Journal of the Association for Computing Machinery*, 7, (1960), 201–214.

[dev89] DEVADAS, S. «Optimal layout via Boolean satisfiability.» En *IEEE International Conference on Computer-Aided Design (ICCAD-89)*. 1989, 294–297.

[dir61] DIREAC, G. A. «On rigid circuit graphs.» *Abh. Math. Sem. Univ. Hamburg*, 25, (1961), 71–76.

[DG84] DOWLING, W.F., y J.H. GALLIER. «Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae.» *J. Log. Program.*, 1, nº 3, (1984), 267–284.

[edm65] EDMONDS, J. R. «Paths, trees and flowers.» *Canad. J. Math.*, 17, (1965), 449–467.

[EG94] EVEN, S., y G. GRANOT. «Grid layouts of block diagrams–bounding the number of bends in each connection.» En *Graph Drawing 94* (R. Tamassia, y I. G. Tollis, eds.). Springer-Verlag, Princeton, New Jersey, 1994, tomo 894 de *Lecture Notes in Computer Science*, 64–75. DIMACS International Workshop.

[EIS76] EVEN, S., A. ITAI, y A. SHAMIR. «On the complexity of timetable and multicommodity flow problems.» *SIAM Journal of Computing*, 5, nº 4, (1976), 691–703.

[FG90] FELDMAN, R., y M. C. GOLUMBIC. «Optimization Algorithms for Student Scheduling via Constraint Satisfiability.» *The Computer Journal*, 33, nº 4, (1990), 356–364.

[FW91] FORMANN, M., y F. WAGNER. «A packing problem with applications in lettering of maps.» En *7th ACM Symposium on Computacional Geometry*. ACM Press, 1991, 281–288.

[FG03] FRANCO, J., y A. VAN GELDER. «A perspective on certain polynomial-time solvable classes of satisfiability.» *Discrete Appl. Math.*, 125, nº 2-3, (2003), 177–214.

[FF67] FRÉCHET, M., y K. FAN. *Initiation to Combinatorial Topology*. Weber & Schmidt, Boston, Prindle, 1967.

[FS96] FRIEZE, A. M., y S. SUEN. «Analysis of Two Simple Heuristics on a Random Instance of k-SAT.» *J. Algorithms*, 20, nº 2, (1996), 312–355. URL `citeseer.ist.psu.edu/article/frieze96analysis.html`.

[FG65] FULKERSON, D.R., y O.A. GROSS. «Incidence matrices and interval graphs.» *Pacific J. Math.*, 15, nº 3, (1965), 835–855.

[GS88] GALLO, G., y M. G. SCUTELLA. «Polynomially solvable satisfiability problems.» *Inf. Process. Lett.*, 29, nº 5, (1988), 221–227.

[GJ79] GAREY, M. R., y D. S. JOHNSON. *Computers and Intractability: a guide to the theory of NP–completeness*. Freeman, 1979.

[gar97] GARRIDO, M. A. *Inmersiones ortogonales de grafos en superficies no planas*. Tesis doctoral, Universidad de Sevilla, Dpto. de Matemática Aplicada I, 1997.

[GIM01] GARRIDO, M. A., C. ITURRIAGA, A. MÁRQUEZ, J. R. PORTILLO, P. REYES, y A. WOLFF. «Labeling subway lines.»

En *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*. Springer-Verlag, Christchurch, Nueva Zelanda, 2001, tomo 2223 de *Lecture Notes in Computer Science*, 645–659.

[GM98] GARRIDO, M. A., y A. MÁRQUEZ. «Embedding a graph in the grid of a surface with the minimum number of bends is NP–hard.» En *GD '97: Proceedings of the 5th International Symposium on Graph Drawing* (G. di Battista, ed.). Springer-Verlag, Roma, Italia, 1997, tomo 1353 de *Lecture Notes in Computer Science*, 124–133.

[GMM02] GARRIDO, M. A., A. MÁRQUEZ, A. MORGANA, y J. R. PORTILLO. «Single bend wiring on surfaces.» *Discrete and Applied Mathematics*, 117, nº 1-3, (2002), 27–40.

[gel02] GELDER, A. Van. «Another Look at Graph Coloring via Propositional Satisfiability.» *Discrete Applied Mathematics*, (to appear). (Preliminary version appeared in COLOR02, held in conjunction with CP02, Ithaca, NY).

[gol02] GOLDBERG, E. «Proving Unsatisfiability of CNFs Locally.» *J. Autom. Reasoning*, 28, nº 5, (2002), 417–434.

[gol05] —. «Testing satisfiability of CNF formulas by computing a stable set of points.» *Ann. Math. Artif. Intell.*, 43, nº 1, (2005), 65–89.

[GN02] GOLDBERG, E., y Y. NOVIKOV. «BerkMin: A Fast and Robust SAT Solver.», 2002. URL `citeseer.ist.psu.edu/goldberg02berkmin.html`.

[GM01] GRIMA, C., y A. MÁRQUEZ. *Computational Geometry on Surfaces*. Kluwer Academic Publishers, Dordrecht - Boston - Londres, 2001.

[GPF97] GU, Jun, Paul W. PURDOM, John FRANCO, y Benjamin W. WAH. «Algorithms for the satisfiability (SAT) Problem: A survey.» En *Satisfiability Problem: Theory and applications* (Ding-Zhu Du, Jun Gu, y Panos Pardalos, eds.), American Mathematical Society, 1997, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 19–152.

[GP95] GU, Jun, y Ruchir PURI. «Asynchronous Circuit Synthesis with Boolean Satisfiability.» *IEEE Transactions on Computer-Aided Design*, 14, nº 8, (1995), 961–973. URL `citeseer.ist.psu.edu/gu95asynchronous.html`.

[gut07] GUTIÉRREZ, J. *Resolución del problema SAT en tiempo polinomial*. Proyecto Fin de Carrera, ETSII - Universidad de Sevilla, 2007.

[har72] HARARY, F. *Graph Theory*. Addison Wesley Publishing Company, 1972.

[hay97] HAYES, B. «Cant't Get No Satisfaction.» *American Scientist*. URL `http://www.amsci.org/amsci/issues/Comsci97/`.

[her09] HERBSTRITT, M. *Satisfiability & Verification*. Sudwestdeutscher Verlag fur Hochschulschriften, 2009.

[HHM04] HOANGA, C. T., S. HOUGARDY, F. MAFFRAY, y N. MAHADEV. «On simplicial and co-simplicial vertices in graphs.» *Discrete Applied Mathematics*, 138, nº 1-2, (2004), 117–132.

[HT74] HOPCROFT, J. E., y R. E. TARJAN. «Efficient planarity testing.» *J. Assoc. Comput. Match.*, 21, (1974), 549–568.

[HGW93] HUANG, X., J. GU, y Y. WU. «A Constrained Approach to Multifont Chinese Character Recognition.» *IEEE Trans. Pattern Anal. Mach. Intell.*, 15, nº 8, (1993), 838–843.

[IA86] IMAI, H., y T. ASANO. «Efficient algorithms for geometric graph search problems.» *SIAM J. Comput.*, 15, nº 2, (1986), 478–494.

[imh75] IMHOF, E. «Positioning names on maps.» *The American Cartographer*, 2, nº 2, (1975), 128–144.

[IM82] ITAI, A., y J. MAKOWSKY. «On the complexity of Herbrand's theorem.» *Working Paper 243, Department of Computer Science, Israel Institute of Technology*.

[jev90] JEVONS, W. S. «Pure Logic and Other Minor Works, Pure Logic or the Logic of Quality Apart From the Quantity.» *Macmillan and Co.*, (1890), 4,5.

[knu76] KNUTH, D. «Big omicron and big omega and big theta.» *SIGACT News*, 8, nº 2, (1976), 18–24.

[kul00] KULLMANN, O. «Investigations on autark assignments.» *Discrete Appl. Math.*, 107, nº 1-3, (2000), 99–137.

[kur30] KURATOWSKI, C. «Sur le problème des courbes gauches en topologie.» *Fund. Math.*, 15, (1930), 271–283.

[sgl34] L'ARMÉE, Service Géographique De. *Les Écritures Sur Les Cartes Topographiques*. Imprimerie Du Service Géographique De L'armée, 1934.

[lar92] LARRABEE, T. «Test pattern generation using Boolean satisfiability.» *IEEE Transactions on CAD*, 11, nº 1, (1992), 4–15.

[len90] LENGAUER, T. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner/Wiley & Sons, Stuttgart/Chicester, 1990.

[liu95] LIU, Y. P. *Embeddability in Graphs*. Kluwer Academic Publishers, 1995.

[LMS91] LIU, Y. P., A. MORGANA, y B. SIMEONE. «General theoretical results on rectilinear embeddadility of graphs.» *Acta Math. Appl. Sinica*, 7, nº 2, (1991), 187–192.

[LMS98] —. «A linear algorithm for 2–bend embeddings of planar graphs in the two–dimensional grid.» *Discrete Applied Mathematics*, 81, nº 8, (1998), 69–91.

[LO06] LYNCE, I., y J. OUAKNINE. «Sudoku as a SAT Problem.» En *9th International Symposium on Artificial Intelligence and Mathematics*. 2006.

[maa00] VAN MAAREN, H. «A Short Note on Some Tractable Cases of the Satisfiability Problem.» *Information and Computation*, 158, nº 2, (2000), 125–130.

[MS91] MARKS, J., y S. SHIEBER. «The Computational Complexity of Cartographic Label Placement.» *Inf. Téc. TR-05-91*, Harvard University, 1991. URL `citeseer.ist.psu.edu/marks91computational.html`.

[SS99] MARQUES-SILVA, J.P, y K.A SAKALLAH. «GRASP – A Search Algorithm for Propositional Satisfiability.» *IEEE Transactions on Computers*, 48, nº 5, (1999), 506–521.

[MS77] MEYER, A.R., y M.I. SHAMOS. «Time and Space.» En *Perspectives on Computer Science* (A.K. Jones, ed.). Academic Press, New York, 1977, 125–146.

[MMZ01] MOSKEWICZ, Matthew W., Conor F. MADIGAN, Ying ZHAO, Lintao ZHANG, y Sharad MALIK. «Chaff: Engineering an Efficient SAT Solver.» En *Proceedings of the 38th Design Automation Conference (DAC'01)*. 2001. URL `citeseer.ist.psu.edu/moskewicz01chaff.html`.

[NC88] NISHIZEKI, T., y N. CHIBA. *Planar Graphs: Theory and Algorithms*. Mathematics Sudies. North-Holland, 1988.

[oro94] O'ROURKE, J. *Computational Geometry in C*. Cambridge University Press, 1994.

[PSS01] POON, S. H., C.S. SHIN, T. STRIJK, y A. WOLFF. «Labeling Points with Weights.» En *ISAAC '01: Proceedings of the 12th International Symposium on Algorithms and Computation*. Springer-Verlag, London, UK, 2001, 610–622.

[por02] PORTILLO, J. R. *Problemas de conexiones ortogonales*. Tesis Doctoral, Universidad de Sevilla, Dpto. de Matemática Aplicada I, 2002.

[PS85] PREPARATA, F. P., y M. I. SHAMOS. *Computational Geometry. An Introduction*. Springer-Verlag, 1985.

[pre93] PRETOLANI, D. *Satisfiability and Hypergraphs*. Tesis Doctoral, Università di Pisa, 1993.

[pre96] —. «Hierarchies of Polynomially Solvable Satisfiability Problems.» *Ann. Math. Artif. Intell.*, 17, nº 3-4, (1996), 339–357.

[raf95] RAFFALLI, C. «Alghorithmes pour la logique.» *Inf. téc.*, CNRS – Université Paris 7, Febrero 1995. En francés.

[RCS86] RAGHAVAN, R., J. COHOON, y S. SAHNI. «Single Bend Wiring.» *Journal of Algorithms*, 7, nº 2, (1986), 232–257.

[RB83] RAO, C. D. V. P., y N. N. BISWAS. «On the Minimization of Wordwidth in the Control Memory of a Microprogrammed Digital Computer.» *IEEE Trans. Comput.*, 32, nº 9, (1983), 863–868.

[rey02] REYES, P. *Problemas de Etiquetado: Complejidad Computacional*. Tesis Doctoral, Universidad de Sevilla, Dpto. de Matemática Aplicada I, 2002.

[sai08] SAÏS, L. *Problème SAT: progrès et défis*. Hermes Science Publications, 2008.

[SS03] SAWADA, J., y J.P. SPINRAD. «From a simple elimination ordering to a strong elimination ordering in linear time.» *Information Processing Letters*, 86, nº 6, (2003), 299–302.

[sch78] SCHAEFER, T. J. «The complexity of satisfiability problems.» En *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*. ACM, New York, NY, USA, 1978, 216–226.

[SAF95] SCHLIPF, J., F. ANNEXSTEIN, J. FRANCO, y R. SWAMINATHAN. «On finding solutions for extended Horn formulas.» *Information Processing Letters*, 54, nº 3, (1995), 133–137.

[scu90] SCUTELLA, M. G. «A note on Dowling and Gallier's top-down algorithm for propositional Horn satisfiability.» *Journal Logic Programming*, 8, nº 3, (1990), 265–273.

[SLM92] SELMAN, Bart, Hector J. LEVESQUE, y D. MITCHELL. «A New Method for Solving Hard Satisfiability Problems.» En *Proceedings of the Tenth National Conference on Artificial Intelligence* (Paul Rosenbloom, y Peter Szolovits, eds.). AAAI Press, Menlo Park, California, 1992, 440–446. URL citeseer.ist.psu.edu/selman92new.html.

[SS00] SHEERAN, Mary, y Gunnar STÅLMARCK. «A Tutorial on Stålmarck's Proof Procedure for PropositionalLogic.» *Form. Methods Syst. Des.*, 16, nº 1, (2000), 23–58.

[SY01] SHIN, C.S., y T. YANG. «Labeling a rectilinear map with sliding labels.» *International Journal of Computational Geometry & Applications*, I, nº 11, (2001), 167–179.

[SS96] SILVA, João P. Marques, y Karem A. SAKALLAH. «GRASP – a new search algorithm for satisfiability.» En *ICCAD '96: Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, Washington, DC, USA, 1996, 220–227.

[SBV96] STEPHAN, P., R.K. BRAYTON, y A.L. SANGIOVANNI-VINCENTELLI. «Combinational test generation using satisfiability.» *IEEE Transactions on CAD*, 15, nº 9, (1996), 1167–1176.

[SK99] STRIJK, T., y M. VAN KREVELD. «Labeling a rectilinear map more efficiently.» *Information Processing Letters*, 69, nº 1, (1999), 25–30. URL citeseer.ist.psu.edu/article/strijk98labeling.html.

[tam87] TAMASSIA, R. «On embedding a graph in the grid with the minimum number of bends.» *Siam J. Comput.*, 16, nº 3, (1987), 421–444.

[tov84] TOVEY, C.A. «A simplified NP-complete satisfiability problem.» *Discrete Applied Mathematics*, 8, nº 1, (1984), 85–89. elsevier:10.1016/0166-218X(84)90081-7.

[tru82] TRUEMPER, K. «Alpha-balanced graphs and matrices and gf(3)-representability of matroids.» *J. Comb. Theory, Ser. B*, 32, nº 2, (1982), 112–139.

[tru98] —. *Effective Logic Computation*. John Wiley & Sons, 1998.

[tse70] TSEITIN, G. S. «On the complexity of derivation in propositional calculus.» En *Studies in Constructive Mathematics and Mathematical Logic, Part 2* (A. O. Slisenko, ed.), Consultants Bureau, New York, 1970. 115–125.

[val00] DEL VAL, A. «On 2-SAT and Renamable Horn.» En *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press / The MIT Press, 2000, 279–284.

[WR96] WANG, Wei, y C.K. RUSHFORTH. «An adaptive local-search algorithm for the channel-assignmentproblem (CAP).» *IEEE Transactions on Vehicular Technology*, 45, nº 3, (1996), 459–466.

[WM00a] WARNERS, J., y H. VAN MAAREN. «Solving satisfiability problems using elliptic approximations. Effective branching rules.» *Discrete Applied Mathematics*, 107, nº 1-3, (2000), 241–259.

[WM00b] WARNERS, J. P., y H. VAN MAAREN. «Recognition of tractable satisfiability problems through balanced polynomial representations.» *Discrete Appl. Math.*, 99, nº 1-3, (2000), 229–244.

[whi73] WHITE, A. T. *Graphs, Groups and Surfaces*. North-Holland, Amsterdam, 1973.

[WS96] WOLFF, A., y T. STRIJK. «The Map-Labeling Bibliography.», 2007. Http://i11www.ira.uka.de/map-labeling/bibliography/, URL `http://i11www.iti.uni-karlsruhe.de/map-labeling/bibliography/maplab_date.html`.

[WR97] WOOD, R. Glenn, y Rob A. RUTENBAR. «FPGA routing and routability estimation via Boolean satisfiability.» En *FPGA '97: Proceedings of the 1997 ACM fifth international symposium on Field-programmable gate arrays*. ACM, New York, NY, USA, 1997, 119–125.

[YS03] YATO, T., y T. SETA. «Complexity and Completeness of Finding Another Solution and Its Application to Puzzles.» *IEICE Trans. Fundamentals*, E86-A, nº i5, (2003), 1052–1060.

[yen94] YEN, H. C. «On Multiterminal Single Bend Wirability.» *IEEE Trans. on Computer–Aided Design of Integrated Circuits and Systems*, 13, nº 6, (1994), 822–826.

[zam05] ZAMBELLI, Giacomo. «A polynomial recognition algorithm for balanced matrices.» *J. Comb. Theory Ser. B*, 95, nº 1, (2005), 49–67.

[ZS96] ZHANG, H., y M. E. STICKEL. «An efficient Algorithm for Unit Propagation.» En *Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics (AI-MATH'96)*. Fort Lauderdale (Florida USA), 1996. URL `citeseer.ist.psu.edu/zhang96efficient.html`.

[zha97] ZHANG, Hantao. «SATO: an efficient propositional prover.» En *Proceedings of the International Conference on Automated Deduction (CADE'97), volume 1249 of LNAI*. 1997, 272–275. URL `citeseer.ist.psu.edu/zhang97sato.html`.

Índice alfabético