Un algoritmo polinomial para problemas de satisfacibilidad*

J.R.Portillo** J.I.Rodrigues ***

Resumen

En este trabajo presentamos una nueva clase de fórmulas lógicas, (PURL), cuya satisfacibilidad puede determinarse en tiempo polinomial usando el algoritmo ELIMINALITERALES. Este algoritmo utiliza el concepto de *literal eliminable* y el hecho de que una fórmula proposicional es equivalente a la obtenida eliminando sus literales eliminables.

La clase PURL aparece al estudiar un problema de dibujo ortogonal de grafos, que podemos reducir a un subconjunto del problema SAT. También son reducibles a fórmulas pertenecientes a esta clase otros problemas de origen geométrico.

1. Introducción

El problema *Satisfacibilidad* (SAT) es un problema fundamental en Lógica Matemática y Teoría de la Computación, con múltiples aplicaciones prácticas en diversos campos [4]. Presentamos a continuación unas cuantas definiciones previas al enunciado formal del problema:

Sea $U = \{u_1, u_2, \ldots, u_n\}$ un conjunto de *variables booleanas*. Una *asignación de verdad* para U es una función $t : U \longrightarrow \{0, 1\}$. Si $t(u) = 1$ decimos que u es *verdadera* bajo t; si $t(u) = 0$ decimos que u es *falsa*. Si u es una variable de U entonces u y $\overline{u}$ son *literales* sobre U. El literal u es verdadero bajo t si y sólo si

*Parcialmente subvencionado por PAI FQM-0164, BFM 2001-2474 y FSE, PRODEP 5.3/ALG/194.019/01

**Departamento de Matemática Aplicada 1, Universidad de Sevilla, e-mail: josera@us.es

***Universidade do Algarve, e-mail: jirodrig@ualg.pt

la variable u es verdadera bajo t; el literal $\overline{u}$ es verdadero si y sólo si la variable u es falsa.

Una *cláusula* sobre U es un conjunto de literales sobre U. Representa la disyunción de dichos literales y se *satisface* bajo una asignación de verdad si, y sólo si, al menos uno de sus miembros es verdadero para la asignación. Una *fórmula proposicional en forma normal conjuntiva* o CNF-*fórmula* $S = (C, U)$ es una colección C de cláusulas sobre un conjunto de variables U y representa la conjunción de de dichas cláusulas. Una CNF-fórmula S se *satisface* si y sólo si existe alguna asignación de verdad t para U que satisface simultáneamente todas las cláusulas de C. Tal asignación de verdad se llama *asignación de verdad satisfactoria* para S y podemos representar este hecho como $S(t) = 1$. Diremos en este caso que la CNF-fórmula S es *satisfacible* y en el caso contrario que es *falsable*.

El enunciado del problema SAT queda como sigue:

SATISFACIBILIDAD (SAT) [1]
Entrada: Una CNF-fórmula S.
Pregunta: ¿Existe una asignación de verdad para U que satisfaga C?

Dado que el problema es NP-completo si no se incluyen restricciones en los conjuntos de variables y/o cláusulas, se han propuesto restricciones para clases particulares de fórmulas proposicionales. Un buen resumen es el *survey* de Gu *et al.* [7]. A partir de estos trabajos, se han encontrado algunas clases resolubles en tiempo polinomial: 2-SAT, *Horn*, *q-Horn*, *extended-Horn* y SLUR [3].

Presentaremos en este trabajo una nueva clase de fórmulas proposicionales resolubles en tiempo polinomial. La clave de este resultado son el nuevo concepto de *literal eliminable* y el conocido algoritmo PROPUNIT (*Propagación de cláusulas unitarias*), el cual es lineal [2]. El estudio de esta nueva clase viene motivado por la resolución del problema EOSC [8], abierto hasta ahora, que será presentado en la sección 3.

Dada una fórmula proposicional S, se define el 1-*entorno* $Nb(c)$ de una cláusula c de C como el conjunto de todas las asignaciones de verdad que hacen verdadero uno y sólamente uno de los literales de c. Si la fórmula es satisfacible, entonces existe una solución (una asignación de verdad t verificando $S(t) = 1$) que pertenece al 1-entorno de una cláusula [6].

Dada una fórmula proposicional $S = (C, U)$ y una cláusula c en C, denotamos por $Nb(l, c)$ al conjunto de las asignaciones de verdad del 1-entorno de c para las cuales el literal $l \in c$ es verdadero. Asimismo, diremos que un literal l

en esa cláusula es *eliminable* si cualquier asignación de verdad en $Nb(l, c)$ hace falsa la CNF-fórmula S. Una cláusula con un literal eliminable puede simplificarse eliminando ese literal. Debido a la NP-completitud de SAT, detectar los literales eliminables de una fórmula proposicional es un problema NP-completo.

El algoritmo *Propagación de cláusulas unitarias* (PROPUNIT) se utiliza usualmente para resolver instancias del problema 2-SAT. En cada paso, resuelve una cláusula unitaria con el resto de la CNF-fórmula. Se detiene cuando no existen más cláusulas, cuando no quedan cláusulas unitarias o cuando obtiene una cláusula vacía. Presentamos a continuación una versión adaptada del algoritmo tomada de Dalal and Etheringthon [2]. También hemos adaptado la notación presentada en ese trabajo: $(\{\}, U)$ representa una CNF-fórmula cuyo conjunto de cláusulas es vacío, la cual es satisfacible y $(\{\square\}, U)$ denota una CNF-fórmula con una cláusula vacía, la cual no es satisfacible.

Algoritmo 1.1. PROPUNIT

```
Input: Una CNF-fórmula S = (C, U).
Output: Una CNF-fórmula sin cláusulas unitarias.

mientras C tenga una cláusula unitaria
        selecciona una cláusula unitaria {l}
        añade el literal l a la lista L
        C := {c ∈ C : l ∉ c}
        C := {c − {l̄} : c ∈ C}
        si C contiene una cláusula vacía entonces C := {□}
toma para el conjunto V las variables de los literales de L
devuelve S' = (C, U − V) y L
```

Sea S una CNF-fórmula y $c = \{l_1, \ldots, l_k, \ldots, l_q\}$ una cláusula de S. Denotamos por $L(l_k, c)$ al conjunto de cláusulas $\{\{\overline{l_1}\}, \ldots, \{\overline{l_{k-1}}\}, \{l_k\}, \{\overline{l_{k+1}}\}, \ldots, \{\overline{l_q}\}\}$. Obviamente, cualquier asignación de verdad sobre U que satisfaga $(L(l_k, c), U)$ pertenece a $Nb(l_k, c)$.

Un literal l de una cláusula c se denomina *p-eliminable* si PROPUNIT$(C \cup L(l, c), U)$ devuelve $(\{\square\}, U - V)$. Es fácil deducir que un literal p-eliminable es eliminable, pero el recíproco no es cierto en general. Sin embargo, sí se verifica en fórmulas proposicionales obtenidas a partir de determinados problemas geométricos. Definimos entonces la clase PROPUNIT *Removable Literals* (PURL) como el conjunto de CNF-fórmulas para las cuales todo literal eliminable es p-eliminable.

En la Sección 2 presentamos el algoritmo EliminaLiterales, que determina si una CNF-fórmula perteneciente a la clase PURL es satisfacible o no. Este algoritmo detecta y elimina todos los literales p-eliminables de una CNF-fórmula en tiempo polinomial. Por lo tanto, si la CNF-fórmula está en la clase PURL, entonces el resultado de ejecutar el algoritmo es una CNF-fórmula sin literales p-eliminables, en cuyo caso la CNF-fórmula es satisfacible o una CNF-fórmula con una cláusula vacía, en cuyo caso no lo es. Se presenta también una condición suficiente para que una fórmula proposicional pertenezca a la clase PURL.

La sección 3 está dedicada a mostrar brevemente y sin demostraciones algunas aplicaciones de este método a la resolución en tiempo polinomial de dos problemas geométricos que permanecían abiertos.

2. La clase PropUnit *Removable Literals*

Sea $S = (C, U)$ una CNF-fórmula y l un literal eliminable de alguna cláusula c de C. La CNF-fórmula resultante de eliminar el literal l de la cláusula c es equivalente (en el sentido de que es satisfecha por las mismas asignaciones de verdad) a la CNF-fórmula S.

Lema 2.1. *Sea $S = (C, U)$ una CNF-fórmula y $l \in c_0$ un literal eliminable. Sea $C' = (C - \{c_0\}) \cup (\{c_0 - \{l\}\})$. Se verifica que $S' = (C', U)$ y S son equivalentes.*

Demostración. Sea t una asignación de verdad que satisface S y por lo tanto todas sus cláusulas. Como $t \notin Nb(l, c_0)$, hay por lo menos un literal de c_0 distinto de l, verdadero bajo t. Por lo tanto la cláusula $c_0 - \{l\}$ y la fórmula S' se satisfacen bajo la asignación de verdad t. Obviamente que cualquier asignación de verdad que satisface S' también satisface S. □

Repitiendo el proceso de detectar y eliminar todos los literales eliminables en cada cláusula obtendremos una CNF-fórmula sin literales eliminables o con una cláusula vacía. En el primer caso la CNF-fórmula es satisfacible y en el segundo es falsable; y por lo tanto lo mismo ocurre con S. Desgraciadamente, detectar un literal eliminable es un problema NP-completo.

Un literal puede ser eliminable pero no p-eliminable. Estudiaremos a continuación la clase de las CNF-fórmulas para las que todos los literales eliminables son p-eliminables (la clase PropUnit *Removable Literals*, que denotaremos por brevedad PURL). Para ello utilizaremos el algoritmo EliminaLiterales:

Algoritmo 2.2. EliminaLiterales

```
Input: Una CNF-fórmula S = (C, U).
Output: Una CNF-fórmula simplificada sin literales p-eliminables.

repetir
   remlitfree:=1
   para cada c ∈ C
      para cada l ∈ c
         (S', L) :=PropUnit(C ∪ L(l, c), U)
         si S' contiene una cláusula nula
            entonces c := c − {l}, remlitfree:=0
                     si c = □ entonces C := {□}
         si el conjunto de clásulas de S' es vacío entonces C := {}
hasta que (C = {}) o (C = {□}) o (remlitfree=1)
devuelve S.
```

El algoritmo explora cada cláusula buscando y eliminando todos los literales p-eliminables. Se detiene cuando todos los literales de una cláusula han sido eliminados o cuando no quedan más literales p-eliminables en la CNF-fórmula S. Como PropUnit es lineal y se ejecuta una vez por cada literal en cada cláusula, el algoritmo EliminaLiterales se ejecuta en tiempo polinomial en el número de cláusulas y de literales.

Para cualquier CNF-fórmula de la clase PURL se verifica que si la salida del algoritmo EliminaLiterales no contiene ninguna cláusula vacía, entonces la CNF-fórmula es satisfacible. Para cualquier CNF-fórmula, si la salida contiene una cláusula vacía, la CNF-fórmula es falsable. Además, EliminaLiterales devuelve una CNF-fórmula sin ningún literal p-eliminable. La pregunta es ahora si queda algún literal eliminable. El Teorema 2.3 proporciona una condición para que esto no suceda. Por lo tanto, las fórmulas que verifiquen esa condición pertenecerán a la clase PURL.

Dada una CNF-fórmula S, PropUnit(S) proporciona una fórmula modificada cuyas cláusulas provienen de la fórmula original. Llamemos C^* al conjunto de cláusulas modificadas (y no eliminadas) por el algoritmo.

Aplicando sistemáticamente el algoritmo PropUnit($C \cup L(l, c), U$) se verifica que si el subconjunto C^* es siempre vacío se puede obtener la satisfacibilidad o falsabilidad de la fórmula original en tiempo polinomial ($O(|C|)$). De hecho, este

es un sistema de resolución de fórmulas de la clase 2-SAT. El teorema siguiente amplía este resultado:

Teorema 2.3. *Sea S una* CNF*-fórmula sin literales p-eliminables. Si aplicando el algoritmo* PROPUNIT$(C \cup L(l,c), U)$ *a cada $c \in C$ y a cada $l \in c$, se satisface la condición $|C^*| \leq 1$, entonces S no posee literales eliminables.*

Demostración. Por reducción al absurdo: Supongamos que existe un literal $l_0 \in c_0$ eliminable y por lo tanto, cualquier asignación $t \in Nb(l_0, c_0)$ hace falsa la fórmula S.

Aplicando el algoritmo PROPUNIT$(C \cup L(l_0, c_0), U)$ se obtiene una lista de literales $\mathcal{L}$ y una fórmula $S' = (C', U')$, sin cláusulas nulas.

Reiteradamente, se aplica PROPUNIT$(C' \cup L(l, c^*), U')$ añadiendo la lista de literales devuelta a $\mathcal{L}$, hasta que la fórmula devuelta sea vacía. c^* es la cláusula modificada del conjunto C^* si $|C^*| = 1$ o una cláusula de C' en el caso contrario. En ningún paso la fórmula devuelta por el algoritmo contiene alguna cláusula nula, pues en ese caso el literal $l \in c^*$ sería p-eliminable en C' y en C.

El absurdo es que cualquier asignación que hace verdaderos todos los literales de $\mathcal{L}$ pertenece al 1-entorno $Nb(l, c)$ y satisface la fórmula S. □

Corolario 2.4. *Si S es una* CNF*-fórmula sin literales p-eliminables en las mismas condiciones que en el Teorema 2.3, entonces S es satisfacible.*

Demostración. Es suficiente observar que cualquier asignación de verdad que haga verdaderos todos los literales de cualquier lista construida como en la demostración del Teorema 2.3, satisface la CNF-fórmula. □

3. Resolución de problemas geométricos

3.1. US-S(m)-4P

Este es un problema de *etiquetado (map labeling)*. Formalmente:

US-S(m)-4P:
Entrada: Un conjunto de puntos $\{p_1, p_2, \ldots, p_n\}$ sobre la bisectriz del primer cuadrante y un conjunto de cuadrados con los lados paralelos a los ejes coordenados $\{s_1, s_2, \ldots, s_n\}$ de m tamaños diferentes.

Pregunta: ¿Pueden colocarse los cuadrados sin que se solapen de forma que cada punto p_i esté en una esquina del cuadrado s_i?

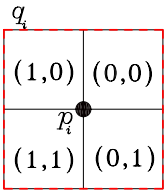

Figura 1: Codificación de las posiciones de un cuadrado con el punto en una esquina

Hay cuatro posibles posiciones para cada cuadrado, que codificamos mediante dos variables booleanas (l_i, b_i). La Figura 1 muestra su codificación, donde $l = 1$ si el cuadrado está a la izquierda del punto y $b = 1$, si está abajo. Ordenando los puntos mediante su posición sobre la recta (por ejemplo de abajo hasta arriba) y etiquetandolos, $p_i < p_j$ si $i < j$, para cada par dos puntos hay cuatro posibles interacciones entre los cuadrados q_i y q_j correspondientes, como se muestra en la Figura 2. Cada una de estas configuraciones da lugar a una CNF-fórmula elemental (Cuadro 1) lo que permite reducir cada instancia del problema US-S(m)-4P a una instancia equivalente del problema SAT.

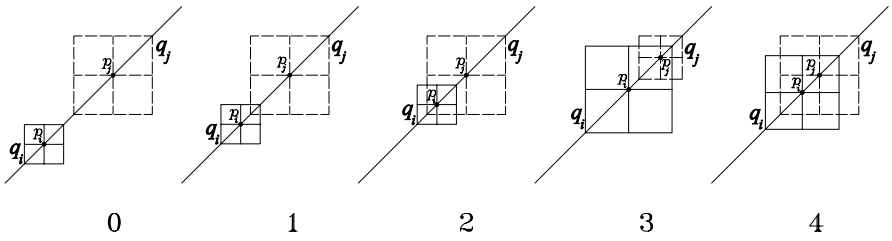

Figura 2: Las cinco posiciones fundamentales de los cuadrados para cada par de puntos

Lema 3.1. *Sea $S = (C, U)$ una* CNF*-fórmula de una instancia del problema* US-S(m)-4P*. Si C no incluye ninguna* CNF*-fórmula elemental del tipo 4 (ver Cuadro 1) entonces la repuesta del problema* US-S(m)-4P *es afirmativa.*

Demostración. Como C solo contiene cláusulas del tipo 1, 2 y/o 3, los pares de literales de cada punto p_i, (l_i, b_i) pertenecen simultáneamente a una misma

0	$[l_i, \bar{l}_i]$
1	$[l_i, b_i, \bar{l}_j, \bar{b}_j]$
2	$[\bar{l}_j, \bar{b}_j]$
3	$[l_i, b_i]$
4	$[l_i, b_i], [l_i, \bar{b}_j], [b_i, \bar{l}_j], [\bar{l}_j, \bar{b}_j]$

Cuadro 1: Fórmulas proposicionales elementales con las cuales se reducen las entradas del problema US-S(m)-4P a SAT. (Ver Fig. 2)

cláusula y son ambos afirmados o ambos negados. Por lo tanto, las asignaciones de verdad con $l_i = 1$ y $b_i = 0$ o $l_i = 0$ y $b_i = 1$ satisfacen S. □

Una cláusula $c \in C$ es *redundante* si existe alguna otra cláusula $c' \in C$ tal que $c' \subset c$. Las cláusulas redundantes pueden eliminarse de la CNF-fórmula obteniendo una CNF-fórmula equivalente. Asumiremos en los siguientes resultados que las fórmulas proposicionales no poseen cláusulas redundantes.

Lema 3.2. *Sea $S = (C, U)$ una CNF-fórmula de una instancia del problema US-S(m)-4P y $c = \{l_i, b_i, \bar{l}_j, \bar{b}_j\}$ una cláusula de C. Entonces entre los puntos p_i y p_j no existen otros intermedios.*

Demostración. Por reducción al absurdo: Supongamos que existe un punto p_k tal que $p_i < p_k < p_j$.

Dado que $c = \{l_i, b_i, \bar{l}_j, \bar{b}_j\}$ es una cláusula de S, entonces p_i no está en el cuadrado q_j y p_j no está en q_i (ver posición 1 en la Figura 2). Si p_k estuviera en el cuadrado q_i, la fórmula elemental de q_i y q_k seria del tipo 3 o 4 (Cuadro 1). Pero en cualquier de los casos, c seria una cláusula redundante, lo que es absurdo porque S no contiene cláusulas redundantes.

Se llega a la misma conclusión si p_k no está en el cuadrado q_i (porque entonces esta en q_j) y por lo tanto el punto p_k no puede existir. □

Lema 3.3. *Sea $S = (C, U)$ una CNF-fórmula de una instancia del problema US-S(m)-4P y S' la fórmula devuelta por el algoritmo* PROPUNIT$(C \cup L(l, c), U)$*, con $c \in C$ y $l \in c$ arbitrarios. Entonces existen fórmulas $S'_a = (C'_a, U'_a)$ y $S'_b = (C'_b, U'_b)$ tales que $C' = C'_a \cup C'_b$ y $U'_a \cap U'_b = \emptyset$*

Demostración. Sean $(S', \mathcal{L})$ la salida de PROPUNIT$(C \cup L(l,c), U)$ y p_i un punto con al menos un literal en c. Consideremos los siguientes conjuntos de cláusulas $C_a^{'} = \{c' \in C' :$ algún literal de c' tiene índice $j < i\}$ y $C_b^{'} = \{c' \in C' :$ algún literal de c' tiene índice $j > i\}$ (Nótese que las variables con literales en $\mathcal{L}$ no están en U').

Si hubiera una cláusula $d \in C'$ que estuviera simultáneamente en $C_a^{'}$ y en $C_b^{'}$, sería de una de las formas siguientes: (a) $d = \{l_r, b_r, \bar{l}_s, \bar{b}_s\}$, (b) $d = \{l_r, \bar{b}_s\}$ o (c) $d = \{b_r, \bar{l}_s\}$, con $r < i < s$. El caso (a) no puede darse por el Lema 3.2. Las cláusulas de los casos (b) o (c) se producirían por una interacción entre q_r y q_s del tipo 4. En ese caso q_r y q_i producirían fórmulas elementales de los tipos 3 o 4 y q_i y q_s, de los tipos 2 o 4. Pero todas las combinaciones son iguales al caso en que todas las formulas elementales son simultáneamente del tipo 4. Analizando esta fórmula con posibles literales del punto p_i en $\mathcal{L}$ se concluye que no puede existir la cláusula d. □

Lema 3.4. *Sea $S = (C, U)$ una* CNF*-fórmula de una instancia del problema* US-S(m)-4P *y $c = \{l_i, b_i, \bar{l}_j, \bar{b}_j\}$ una de sus cláusulas. Además, si existen dos puntos p_h y p_k, $p_h < p_i < p_j < p_k$, entonces no hay en S ninguna cláusula con literales de p_i y p_k juntos ni tampoco de p_h y p_j.*

Demostración. Como los puntos p_i y p_k no pertenecen al cuadrado q_i, si p_i esta en el cuadrado q_k las interacciones entre q_i y q_k son del tipo 2 (verificando el resultado) y si p_i esta en q_k, son del tipo 0 o 1. Pero en este caso la correspondiente cláusula no pertenece a S por el Lema 3.2.
Para los puntos p_h y p_j, la demostración es idéntica. □

Lema 3.5. *Sea $S = (C, U)$ una* CNF*-fórmula de una instancia del problema* US-S(m)-4P *y $S_a^{'}$ y $S_b^{'}$ una partición de la fórmula devuelta por* PROPUNIT$(C \cup L(l,c), U)$*, en las mismas condiciones que el Lema 3.3.*
Entonces el conjunto de las cláusulas modificas (y no removidas) en el algoritmo, $|C^*| \leq 2$ y además cada uno de los subconjuntos de cláusulas $C_a^{'}$ y $C_b^{'}$ contiene a lo sumo una cláusula modificada.

Demostración. Sean c' y d' dos cláusulas modificadas (y no removidas) por el algoritmo, que sin pérdida de generalidad suponemos en $C_a^{'}$. Entonces, las cláusulas originales son del tipo 1, $c = \{l_k, b_k, \bar{l}_r, \bar{b}_r\}$ y $d = \{l_s, b_s, \bar{l}_t, \bar{b}_t\}$, $k < r < s < t < i$ (atendiendo al Lema 3.2). Por el Lema 3.4, no existe ninguna cláusula conteniendo simultáneamente literales de p_k y p_s ni tampoco de p_r y p_t, por lo cual

no puede existir propagación de literales a la cláusula c. El caso $r = s$ tampoco puede darse.

Se concluye entonces que c y d no pueden ser simultáneamente modificadas por el algoritmo PROPUNIT empezando por una cláusula con literales del punto p_i. □

Lema 3.6. *Sea $S = (C, U)$ una* CNF*-fórmula de una instancia del problema* US-S(m)-4P *y S' (sin cláusulas nulas) la correspondiente fórmula sin literales* p*-eliminables, devuelta por el algoritmo* ELIMINALITERALES*(S). Entonces S' tampoco contiene literales eliminables.*

Demostración. Aplicando el algoritmo PROPUNIT($S' \cup L(l, c)$) la fórmula devuelta satisface las condiciones del Lema 3.5. Por lo tanto, el resultado del Teorema 2.3 (y el esquema de su demostración) se aplica a cada una de las componentes de la partición de la fórmula devuelta y se repite hasta obtener una asignación que satisface S' (y por lo tanto S por el Lema 2.1). Como $c' \in C'$ y $l \in c'$ son arbitrarios S' no puede contener literales eliminables. □

El lema anterior muestra que para cualquier instancia del problema US-S(m)-4P la satisfacibilidad de la correspondiente fórmula proposicional puede ser establecida por el algoritmo ELIMINALITERALES. Además, se queda demostrado que eliminando todos los literales p-eliminables se obtiene una formula equivalente sin literales eliminables y por lo tanto todos los literales eliminables son p-eliminables. Como consecuencia la formula pertenece a la classe polinomial PURL lo que permite el resultado siguiente:

Teorema 3.7. *Todas las* CNF*-fórmulas obtenidas de instancias del problema* US-S(m)-4P *pertenecen a la classe* PURL*. Por lo tanto, el problema* US-S(m)-4P *es resoluble en tiempo polinomial.*

3.2. Emparejamiento ortogonal simple en el cilindro (EOSC)

EMPAREJAMIENTO ORTOGONAL SIMPLE EN EL CILINDRO (EOSC) [8]
Entrada: Conjunto de pares de puntos sobre el cilindro.
Pregunta: ¿Existe un trazado ortogonal simple para cada par de de puntos sin intersecciones entre ellos?

El problema del emparejamiento ortogonal simple plano puede ser resuelto en tiempo polinomial [9]. En superficies genéricas, el problema es NP-completo [5].

La complejidad en superficies particulares era un problema abierto. La diferencia principal entre superficies es el nœmero de trazados ortogonales que pueden unir cada par de puntos: dos en el plano, cuatro en el cilindro (Figura 3) y ocho en el toro y superficies de género superior. En el cilidro por lo tanto, podemos codificar cada par de puntos con un par de variables booleanas. Usando éstas, es posible reducir cada instancia de EOSC a una instancia de una subclase de SAT [8].

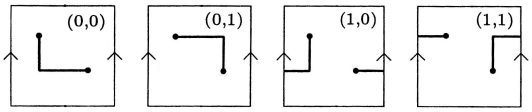

Figura 3: Cuatro trazados ortogonales simples de un par de puntos en el cilindro.

Para este estudio, se han desarrollado una nueva codificación, identificando todas las interacciones entre tríos de pares de puntos. En ella, el puesto de los literales en las cláusulas puede estar ocupado por una conjunción entre literales y sólamente se utiliza una variable de cada dos asociadas a un par de puntos. Las *cláusulas generalizadas* obtenidas son de la forma:

$$[u],\ [u, v],\ [u, v \wedge w, \overline{v} \wedge \overline{w}],\ [u, \overline{v} \wedge w, v \wedge \overline{w}]$$

También se define un grafo asociado a la fórmula proposicional S así obtenida. Los vértices corresponden a los nuevos *literales generalizados*. Dos vértices son adyacentes si no pueden ser satisfechos simultáneamente. En este grafo, un literal generalizado p-eliminable puede identificarse mediante técnicas que nos llevan a concluir (junto con el Teorema 2.3) los siguientes resultados:

Lema 3.8. *Sea S la* CNF*-fórmula de una instancia del problema* EOSC*. Ejecutando repetidas veces el algoritmo* PROPUNIT$(S \cup L(l, c))$*, C^* y siendo $S' = (C', U')$ la salida del algoritmo, se verifica una de las propiedades siguientes:*
a) $|C^*| \leq 1$
b) $|C^*| = 2$ *y existe una partición de C' en dos subconjuntos C'_a y C'_b cuyos literales pertenecen conjuntos de variables disjuntos. Además cada uno de los subconjuntos C'_a y C'_b contienen a lo sumo una cláusula modificada.*

Teorema 3.9. *Las fórmulas proposicionales del problema* EOSC *pertenecen a la clase* PURL *y por lo tanto el problema es resoluble en tiempo polinomial.*

Referencias

[1] 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.

[2] Dalal, M., y D.W. Etheringthon. «A hierarchy of tractable satisfiability problems.» *Information Processing Letters*, 44, nº 4, (92), 173–180.

[3] Franco, J., y A. van Gelder. «A perspective on certain polynomial-time solvable classes of satisfiability.» *Disc. Appl. Math.*, 125, (2003), 177–214.

[4] Garey, M. R., y D. S. Johnson. *Computers and Intractability: a guide to the theory of NP–completeness*. Freeman, 1979.

[5] Garrido, M. A., A. Márquez, A. Morgana, y J. R. Portillo. «Single bend wiring on surfaces.» *Discrete Appl. Math.*, 117, (2002), 27–40.

[6] Goldberg, Eugene. «Proving unsatisfiability of CNFs locally.» En *Elec. Notes in Disc. Math.*, 9 (H. Kautz y B. Selman, eds.). Elsevier, 2001.

[7] Gu, J., P. Purdom, J. Franco, y B. Wah. «Algorithms for the Satisfiability (SAT) Problem: a Survey.», 1996.

[8] Portillo, J. R. *Problemas de conexiones ortogonales*. Tesis Doctoral, Universidad de Sevilla, Dpto. de Matemática Aplicada I, 2002.

[9] Raghavan, R., J. Cohoon, y S. Sahni. «Single Bend Wiring.» *Journal of Algorithms*, 7, (1986), 232–257.