Estudio del problema de la planificación de objetos móviles sin colisiones con obstáculos móviles
Study of the problem of collision-free mobile objects planning under the presence of mobile obstacles

Trabajo de Fin de Grado
Curso 2023–2024

Autor
Daniel María Carreño López

Director
Ismael Rodríguez Laguna
Fernando Rubio Diez

Doble Grado en Ingeniería Informática - Matemáticas
Facultad de Informática
Universidad Complutense de Madrid

Estudio del problema de la planificación de objetos móviles sin colisiones con obstáculos móviles
Study of the problem of collision-free mobile objects planning under the presence of mobile obstacles

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Daniel María Carreño López

Director
Ismael Rodríguez Laguna
Fernando Rubio Diez

Convocatoria: *Septiembre 2024*

Doble Grado en Ingeniería Informática - Matemáticas
Facultad de Informática
Universidad Complutense de Madrid

13 de septiembre de 2024

Agradecimientos

A todas las personas cercanas que me han apoyado mientras realizaba este trabajo, como mi familia y amigos, y en especial a mis tutores Ismael y Fernando que lo han hecho posible.

Resumen

Estudio del problema de la planificación de objetos móviles sin colisiones con obstáculos móviles

En este trabajo consideramos el problema de desplazar dinámicamente varios objetos por las aristas y vértices de un grafo valorado bidireccional en el que hay también obstáculos móviles, sin que los objetos sufran riesgo de colisionar entre sí ni con los obstáculos. Mostramos que averiguar si esto es posible se trata de un problema PSPACE-duro. También discutimos cómo podría llegar a demostrarse que es un problema en PSPACE. Estudiamos también varias variaciones de dicho problema, algunas de ellas PSPACE-completas.

Palabras clave

PSPACE, PSPACE-completitud, PSPACE-dureza, NP-completitud, Complejidad, Planificación de caminos, TQBF.

Abstract

Study of the problem of collision-free mobile objects planning under the presence of mobile obstacles

In this work, we consider the problem of moving dynamically multiple objects through the edges and vertexes of a weighted undirected graph in which there are mobile obstacles while avoiding any risk of the objects colliding between themselves or with the obstacles. We show that determining whether this is possible is a PSPACE-hard problem. We also discuss how this problem could be shown to be in PSPACE. Additionally, we study the complexity of some variations of said problem, some of them PSPACE-complete.

Keywords

PSPACE, PSPACE-completeness, PSPACE-hardness, Complexity, Path-planning, TQBF.

Índice

Índice de figuras

Capítulo 1

Introducción

El estudio de la complejidad computacional de los problemas es un campo de gran interés para muchos investigadores. Año tras año se conoce a qué clase pertenecen más y más problemas distintos, y aun así parece que seguimos desconociendo detalles sobre cómo se relacionan entre ellas. El hecho de que es un campo del que parece que siempre hay más y más cosas por saber es lo que nos ha motivado a realizar este trabajo, que estudia la complejidad de un pequeño grupo de problemas.

Con estos problemas nos referimos a los de decidir si es posible desplazar un conjunto de objetos por un grafo valorado sin riesgo alguno de que colisionen con un conjunto de obstáculos móviles en el mismo grafo. Nos centraremos en uno en concreto y posteriormente estudiaremos alguna de sus variantes. Trabajos anteriores han estudiado la complejidad de problemas similares pero sin obstáculos, que son NP-duros (Goldreich, 2011), (Carreño López, 2024). Si buscamos problemas con más diferencias que la falta de obstáculos, encontramos una gran cantidad de problemas, algunos NP-completos (Ajay et al., 2022) y otros PSPACE-completos (Hearn y Demaine, 2005). Sin embargo, aún no se ha analizado la complejidad de los problemas que vamos a estudiar en este trabajo, por lo que creemos que tienen un interés teórico.

Estos problemas también tienen interés práctico, ya que podrían modelar desde estrategias en un videojuego hasta rutas de drones que tengan que evitar obstáculos reales. Precisamente por esto, ya se ha estudiado el cómo aproximar el movimiento de objetos evitando obstáculos en espacios euclídeos (Masehian y Katebi, 2007), (Bilbeisi et al., 2015); y esperamos que este trabajo sea inspiración para que también se estudie en espacios más generalizados como los grafos.

En el capítulo 2, comenzamos definiendo el problema principal que vamos a tratar. Tras ello, nos servimos del problema PSPACE-completo *True Quantified Boolean Formula* (TQBF) para demostrar que es un problema PSPACE-duro. Posteriormente, discutimos cómo podría llegar a demostrarse su pertenencia a PSPACE. En el capítulo 3 estudiamos varias variantes de este problema, demostrando algunas propiedades de complejidad de ellas. Finalmente, en el capítulo 4, resumimos los resultados y damos algunas ideas de trabajo futuro.

Como plan de trabajo, en primer lugar hemos buscado y leído diversas publicaciones sobre trabajos relacionados. Posteriormente, hemos pensado qué propiedades tenía nuestro problema y cuáles podíamos demostrar. Ahí ideamos la reducción a partir de TQBF que nos ha permitido demostrar que el problema es PSPACE-duro. Tuvimos que continuar refinando esta reducción para poder demostrar que produce instancias equivalentes a las originales. A continuación, pensamos en posibles maneras de demostrar que el problema pertenecía a PSPACE. Al ver que ninguna de ellas iba a poder ser utilizada a tiempo para este trabajo, nos limitamos a recopilar las más prometedoras como indicación para posibles trabajos futuros.

Durante este proceso, anotamos las variantes que parecían más interesantes para estudiarlas también. Con ellas también pensamos en qué podíamos demostrar y si era posible utilizar las demostraciones anteriores para ello. Finalmente, recopilamos todo el trabajo en esta memoria incluyendo explicaciones y ejemplos para que pueda ser comprendida fácilmente.

Capítulo 2

Problema MnOmO

En este capítulo definiremos el problema que vamos a estudiar, mostraremos que es un problema PSPACE-duro y discutiremos cómo podría llegar a mostrarse su PSPACE-completitud. Llamaremos a este problema "Movimiento de n Objetos con m Obstáculos" o MnOmO.

2.1. Definición

Una instancia de MnOmO consta de los siguientes elementos:

- Grafo bidireccional valorado $G = (V, A)$
- $n, m, t \in \mathbb{N}$
- $o, d : \{1, \ldots, n\} \to V$
- $o_m : \{1, \ldots, m\} \to V$

Representamos el subconjunto de aristas de G de la siguiente forma:

$$A \subseteq V \times V \times \mathbb{N}$$

$$(v_1, v_2, d) \in A \Rightarrow (v_2, v_1, d) \in A$$

Decimos que (v_1, v_2, d) y (v_2, v_1, d) son la misma arista.

En el grafo hay n objetos y m obstáculos.

Una simulación es un conjunto de movimientos de los objetos y obstáculos a lo largo del grafo. Una simulación comienza en el momento 0 y termina en el momento t.

En una simulación se cumplen las siguientes reglas:

1. El objeto i comienza en el vértice $o(i)$ y el obstáculo j en el vértice $o_m(j)$. Llamamos a estos vértices los orígenes del objeto i y del obstáculo j respectivamente.
2. Si un objeto u obstáculo está en un vértice v_1 en un momento $x \in \mathbb{N}$ y existe una arista $a = (v_1, v_2, d) \in A$, el objeto u obstáculo puede moverse a través de a para llegar al vértice v_2 en el momento $x + d \in \mathbb{N}$. Decimos que el objeto está moviéndose por a en el sentido $v_1 - v_2$ en el intervalo de tiempo $(x, x + d) \in \mathbb{R}$.
3. Los obstáculos no pueden comenzar un movimiento desde un vértice v_1 a otro v_2 si v_2 pertenece a $Im(d)$, donde $Im(d)$ es la imagen de la función $d : \{1, \ldots, n\} \to V$.
4. Los objetos y obstáculos solo pueden cambiar su posición de acuerdo con la regla 2.
5. Los objetos y obstáculos no pueden detenerse ni cambiar de sentido en una arista.
6. Los objetos y obstáculos pueden detenerse en un vértice durante cualquier cantidad de tiempo dada por un número natural.
7. Si un objeto u obstáculo se mueve por una arista $a = (v_1, v_2, d)$ en el sentido $v_1 - v_2$ en el intervalo $I_1 \subset \mathbb{R}$, otro objeto u obstáculo se mueve por a en el sentido $v_2 - v_1$ en el intervalo $I_2 \subset \mathbb{R}$, y la intersección $I_1 \cap I_2$ es no vacía, decimos que hay una colisión entre ambos objetos u obstáculos.
8. Si dos obstáculos colisionan entre sí, dejan de estar en el grafo durante el resto de la simulación.

El movimiento de los obstáculos no lo podemos decidir. Sin embargo, siempre que un objeto se encuentre en un vértice, debemos decidir por qué arista debe continuar o si debe mantenerse en dicho vértice una cantidad entera de tiempo. Para ello no solo conocemos la instancia, sino también la posición de todos los objetos y obstáculos en ese momento, pero no en momentos futuros.

Por ejemplo, en el momento 0 no sabremos qué arista van a tomar los obstáculos, pero sí en qué vértice se encuentran. En otros momentos de la simulación, podemos tomar distintas decisiones según si los obstáculos se han desplazado de una manera u otra.

Si planificamos qué queremos que hagan los objetos en cada momento en una simulación y para cada posición de los obstáculos y objetos en dicho momento que sea posible tras haber seguido la planificación desde el momento 0, decimos que tenemos una estrategia. En otras palabras, una estrategia es una "guía" que nos especifica qué decisión tomar en todas las situaciones posibles, siempre y cuando hayamos seguido la estrategia para todas las decisiones desde el momento 0.

El problema pregunta:

¿Existe una estrategia tal que, siguiéndola para cualquier movimiento posible de los obstáculos, en ningún caso haya ninguna colisión objeto-objeto ni objeto-obstáculo,

y al acabar la simulación en el momento t todo objeto se encuentre en su destino?

Una vez que ha quedado definido el problema, finalizamos la sección con algunos comentarios y observaciones sobre el mismo:

- Se puede interpretar que los obstáculos se mueven al azar, o que un "oponente" decide qué aristas toman intentando que nuestros objetos no alcancen su destino. Ambas interpretaciones intuitivas son equivalentes desde el punto de vista del problema, puesto que se pregunta si la estrategia es válida para todas las combinaciones posibles de movimientos de los objetos.

- La regla 3 puede parecer artificial e innecesaria. También da la impresión de simplificar el problema al quitarle opciones a los obstáculos. Sin embargo, una vez que se estudian las implicaciones de que los obstáculos puedan ir a los destinos de los objetos, parece que daría lugar a un problema más sencillo, ya que la mejor "estrategia" de los obstáculos podría ser ir directamente a ellos. De hecho, más adelante discutiremos la posibilidad de que el problema sin esta regla sea NP-completo.

- El propósito de la regla 8 es evitar que los obstáculos se "atraviesen" entre sí ignorando colisiones, ya que el objetivo del problema ya impide que los objetos lo hagan entre sí o con obstáculos. Los resultados que vamos a mostrar sobre el problema seguirían siendo ciertos sin esta regla.

- Solo hay colisiones en aristas lo que, entre otras cosas, implica que dos objetos u obstáculos pueden "cruzarse" siempre que lo hagan en un vértice. Esto también nos simplifica la reducción. Estudiaremos más adelante qué sucede si hay colisiones en los vértices.

- A la arista (v_1, v_2, d) también la representaremos con $v_1 - v_2$, y a un camino que pase por los vértices $v_1, v_2, \ldots, v_n$ en ese orden lo representaremos por $v_1 - v_2 - \cdots - v_n$.

2.2. PSPACE-dureza

Para demostrar que el problema MnOmO es PSPACE-duro, haremos una reducción polinómica del problema *True Quantified Boolean Formula* (TQBF) restringido a fórmulas en forma normal conjuntiva (FNC) en MnOmO.

En esta sección comenzaremos analizando ligeramente el problema TQBF y veremos que puede representarse como un juego de dos jugadores con información perfecta. Posteriormente definiremos la reducción en concreto, acompañada de ejemplos, y finalizaremos demostrando que toda instancia de TQBF es equivalente a la instancia de MnOmO resultante de aplicar la reducción sobre ella. Para esto último utilizaremos que TQBF se puede representar como un juego con un árbol de juego.

2.2.1. TQBF

Una *Quantified Boolean Formula* (QBF) tiene la forma 2.1, donde $Q_1, Q_2, \ldots, Q_k$ son cuantificadores de entre $\exists$ y $\forall$, y $\phi(x_1, x_2, \ldots, x_k)$ es una fórmula booleana sin cuantificadores.

$$Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \phi(x_1, x_2, \ldots, x_k) \tag{2.1}$$

Una instancia del problema TQBF consiste en decidir si una QBF es cierta o falsa. TQBF es un problema PSPACE-completo (Garey y Johnson, 1979). También es un problema PSPACE-completo si limitamos ϕ de manera que tenga que estar en FNC (Schaefer, 1976). En el resto del trabajo asumiremos que las fórmulas ϕ están en FNC, es decir, que son una conjunción de cláusulas disyuntivas. Un ejemplo de QBF donde ϕ está en FNC es 2.2.

$$\forall x_1 \exists x_2 (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \tag{2.2}$$

Un detalle relevante es que el orden en el que aparecen cuantificadas las variables importa. El ejemplo anterior se debe leer como en 2.3. Si apareciese primero $\exists x_2$ y después $\forall x_1$ se leería como en 2.4, y 2.4 es falso mientras que 2.2 es cierto.

$$\forall x_1 \text{ se cumple } (\exists x_2 \text{ tal que } (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)) \tag{2.3}$$

$$\exists x_2 \text{ tal que } (\forall x_1 \text{ se cumple } (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)) \tag{2.4}$$

Una forma de saber si una QBF es cierta consiste en sustituir las variables en orden con todos los valores posibles. Podemos utilizar la veracidad de la fórmula tras haber sustituido variables para averiguar la veracidad de la QBF original. En concreto, al tener un $\forall$ en la primera variable, el QBF original es cierto si y solo si son ciertos los dos QBFs resultantes de sustituir dicha variable por $\top$ y por $\bot$. En cambio, si la primera variable va acompañada de un $\exists$, el original será cierto si y solo si al menos una de las dos sustituciones lo es. Más formalmente, dadas las definiciones 2.5, se cumple 2.6.

$$\begin{aligned} A &= Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \phi(x_1, x_2, \ldots, x_k) \\ A_c &= Q_2 x_2 \ldots Q_k x_k \phi(\top, x_2, \ldots, x_k) \\ A_f &= Q_2 x_2 \ldots Q_k x_k \phi(\bot, x_2, \ldots, x_k) \end{aligned} \tag{2.5}$$

$$\begin{aligned} Q_1 = \forall \Rightarrow A = A_c \wedge A_f \\ Q_1 = \exists \Rightarrow A = A_c \vee A_f \end{aligned} \tag{2.6}$$

Esta propiedad también se cumple en el resto de variables cuando pasan a ser la primera. Más generalmente tiene la forma de 2.7, donde G es una fórmula booleana con cuantificadores o sin ellos, y $G[x/y]$ indica la sustitución de x por y en G.

$$\begin{aligned} F = \forall x G \Rightarrow F = G[x_1/\top] \wedge G[x_1/\ \bot] \\ F = \exists x G \Rightarrow F = G[x_1/\top] \vee G[x_1/\ \bot] \end{aligned} \tag{2.7}$$

Si aplicamos esto a cada una de las k variables de la QBF para finalmente llegar a fórmulas sin cuantificadores ni variables. Por tanto con 2.7 nos basta para convertir la QBF en una fórmula booleana sin variables, aunque para ello es necesario aplicar 2.7 2^{k-1} veces donde k es el número de variables.

Cada vez que sustituimos una variable, nos surgen dos fórmulas nuevas. Por tanto, podemos visualizar este proceso como un árbol binario, donde la raíz corresponde al QBF original, sus hijos a las fórmulas resultantes de sustituir la primera variable, y así sucesivamente hasta que las hojas corresponden a fórmulas sin variables.

En la figura 2.1 podemos ver una representación del árbol descrito. Los hijos izquierdos corresponden a sustituir por $\top$ la primera variable y los derechos a sustituir por $\bot$. En el lateral derecho aparece para cada nivel del árbol el primer cuantificador que hay en las fórmulas de dicho nivel. Dado un nivel i (donde el nivel 1 es el de la raíz), diremos que es un nivel $\exists$ si Q_i es un $\exists$ y que es un nivel $\forall$ de lo contrario. Si un vértice representa una fórmula cierta diremos que es cierto y de lo contrario diremos que es falso.

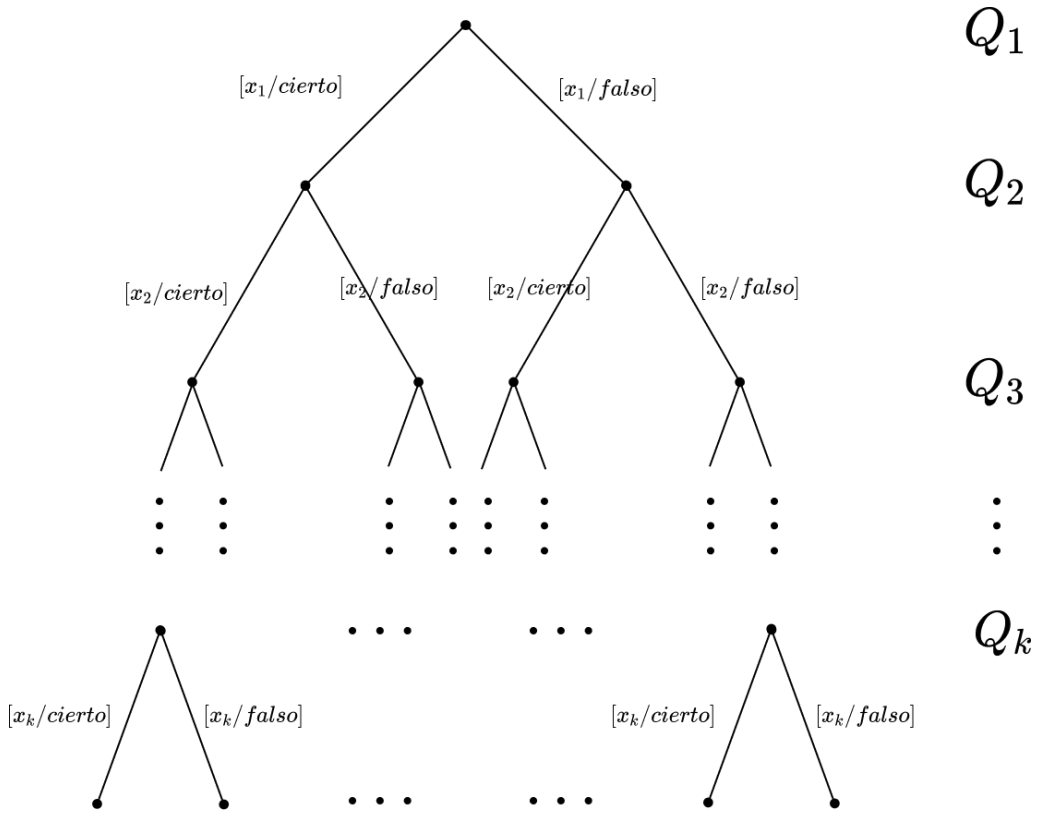

Figura 2.1: Árbol binario

En este árbol, las hojas corresponden a fórmulas sin variables, por lo que podemos saber directamente si son ciertas o falsas. Para saber la veracidad de un vértice

intermedio, necesitamos conocer las veracidades de sus hijos. Con el valor de sus hijos podemos saber su valor siguiendo 2.7, es decir, si está en un nivel $\exists$ se cumple el vértice si se cumple al menos uno de sus hijos, y si está en un nivel $\forall$ se cumple si se cumplen sus dos hijos.

El árbol que hemos descrito tiene mucho en común con lo que se conoce como árbol de juego. Ahora veremos que esto se debe a que TQBF es equivalente a un juego de dos jugadores con información perfecta. Llamaremos a este juego, que describimos a continuación, el juego TQBF.

En este juego un jugador elige el valor de las variables cuantificadas con un $\forall$ y el otro el de las cuantificadas con un $\exists$. Llamaremos al primero de ellos el jugador $\forall$ y al segundo el jugador $\exists$. Al comenzar la partida, el jugador que tenga su primer turno tiene dos opciones: asignar $\top$ o asignar $\bot$ a la primera variable. Qué jugador sea este depende de cuál sea el primer cuantificador. Posteriormente uno de los dos jugadores tiene que asignarle un valor a la siguiente variable según el cuantificador de esta y así sucesivamente. Al ir a tomar una decisión, ambos jugadores conocen las asignaciones que se han hecho en los turnos anteriores. La estructura de las posibles partidas diferentes entre los dos jugadores tiene por tanto la misma estructura que el árbol de la figura 2.1. En este caso, el cuantificador Q_i nos indica de qué jugador es el turno en el nivel i del árbol.

En el juego TQBF el objetivo del jugador $\exists$ es que la fórmula evalúe a $\top$ después de asignar valores a todas las variables. El jugador $\exists$ tiene una estrategia ganadora al comenzar el juego si puede elegir dinámicamente cómo asigna las variables de manera que gane independientemente de las acciones del jugador $\forall$. Se puede estudiar si hay una estrategia ganadora al empezar el juego, que es la raíz del árbol, pero también si la hay tras haber asignado ya algunas variables, lo que corresponde con un vértice intermedio del árbol.

En las hojas es muy fácil saber si hay una estrategia ganadora ya que no queda ninguna decisión que tomar, por lo que solo existe una estrategia posible: no hacer nada. Si el valor de la fórmula en esa hoja, que se halla sustituyendo cada variable por $\top$ o $\bot$ según qué camino haya que seguir desde la raíz, es $\top$, entonces la estrategia es ganadora y de lo contrario no lo es.

Que haya una estrategia ganadora en un vértice no hoja se puede averiguar sabiendo de qué jugador es el turno y si la hay en sus hijos. Si es el turno del jugador $\exists$, basta con que haya una estrategia ganadora para uno de los hijos. En tal caso, la nueva estrategia será escoger $\top$ o $\bot$ para la variable según qué hijo tenga una estrategia ganadora, y luego seguir la estrategia de dicho hijo. Si ambos hijos tienen una estrategia ganadora, tanto escoger $\top$ como $\bot$ y luego seguir la estrategia del hijo correspondiente son estrategias ganadoras.

En el turno del jugador $\forall$, el jugador $\exists$ no puede controlar qué elección se va a tomar. Por tanto es necesario que haya una estrategia ganadora en ambos hijos para que la haya en el padre. Así, la estrategia consistirá en seguir la estrategia ganadora de uno de los dos hijos según elija $\top$ o $\bot$ el jugador $\forall$.

Sabiendo esto es posible recorrer el árbol para averiguar si existe una estrategia

ganadora para el jugador $\exists$. Este método es equivalente al método para averiguar si una QBF es cierta siguiendo 2.7, ya que F corresponde al vértice padre y $G[x/\top]$, $G[x/\ \bot]$ corresponden a sus hijos. Por esto podemos decir que una instancia TQBF es cierta si y solo si existe una estrategia ganadora para el jugador $\exists$ en el juego TQBF correspondiente. Esto nos servirá para tratar TQBF como un juego y para ver el árbol de la figura 2.1 como un árbol de juego, estructura comúnmente utilizada para estudiar juegos (Allis, 1994).

Como ejemplo particular podemos pensar lo que ocurre si todos los cuantificadores son $\forall$. Viendo el problema desde el punto de vista de la lógica, ha de cumplirse la fórmula con todas las combinaciones de valores asignados a las variables. Viendo el problema como un juego, es necesario que el jugador $\forall$ no tenga ninguna manera de asignar las variables de forma que la formula no se cumpla. Se puede ver que las conclusiones son las mismas desde ambas perspectivas. En el árbol de juego hace falta que ninguna hoja corresponda a una fórmula que no se cumpla, ya que las hojas corresponden a todas las combinaciones de valores asignados a las variables.

En el ejemplo contrario, donde todos los cuantificadores son $\exists$, también se puede llegar a la misma conclusión desde ambas perspectivas. En este caso basta con que alguna combinación de valores asignados a las variables haga que la fórmula se cumpla, o equivalentemente que el jugador $\exists$ tenga una manera de escoger los valores para hacer la fórmula cierta. En el árbol de juego esto significa que es necesario que al menos una hoja corresponda a una fórmula cierta.

A partir de este punto hablaremos de TQBF como un juego, donde si existe una estrategia ganadora para el jugador $\exists$, entonces la respuesta a la instancia del problema es afirmativa. Ahora vamos a ver cómo se puede simplificar el árbol de juego para incluir solo la información necesaria.

En los vértices del árbol donde el cuantificador es $\exists$, tomamos nosotros la decisión de qué valor le damos a la variable. Con que una opción nos garantice la victoria, también tenemos la victoria garantizada en nuestro vértice siempre que escojamos esa opción. No tenemos por tanto ninguna necesidad de escoger la otra opción, y por tanto no necesitamos saber si al escoger la otra opción seguimos teniendo una estrategia ganadora o no. Por ello, cada vez que uno de los hijos de un vértice donde es nuestro turno nos garantice la victoria, podemos simplemente "podar" el otro o, en otras palabra, no mostrar ni el hijo ni ninguno de sus descendientes. De esta manera seguimos pudiendo utilizar el árbol para comprobar si existe una estrategia ganadora, pero este tiene menos información extra.

$$\begin{gathered} \exists x_1 \forall x_2 \exists x_3 \phi(x_1, x_2, x_3) \\ \phi(x_1, x_2, x_3) = (x_1 \vee \neg x_1) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \end{gathered} \tag{2.8}$$

Dada la QBF descrita en 2.8, mostramos en la figura 2.2 su árbol de juego sin podar y en la figura 2.3 el mismo árbol podado. En ambas figuras los cuantificadores a la derecha del árbol indican a qué jugador le toca asignar un valor a una variable.

Se puede observar que el árbol podado de la figura 2.3 es más compacto que el no

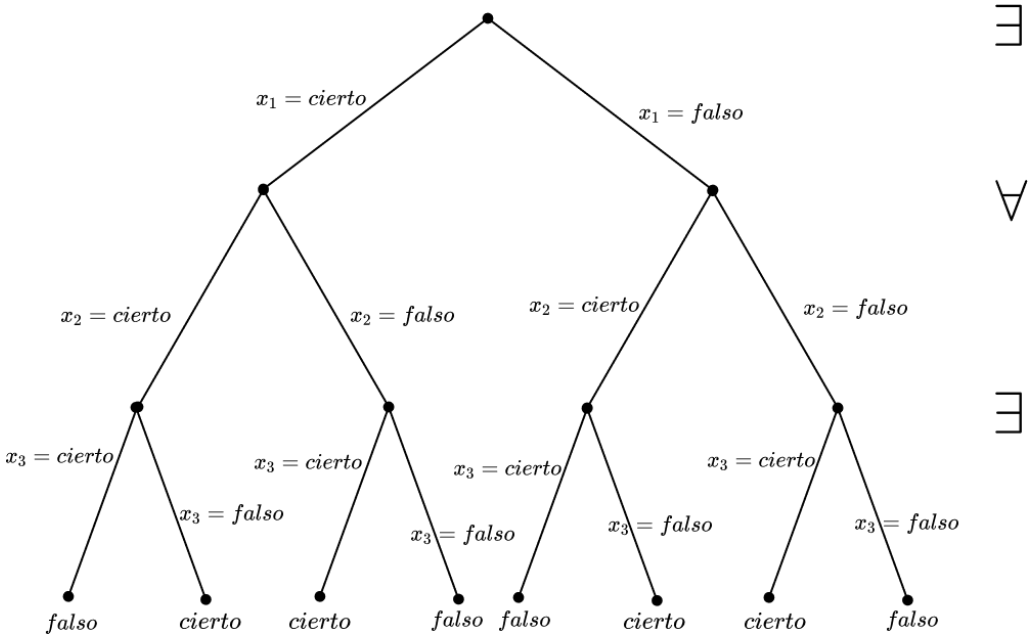

Figura 2.2: Árbol no podado

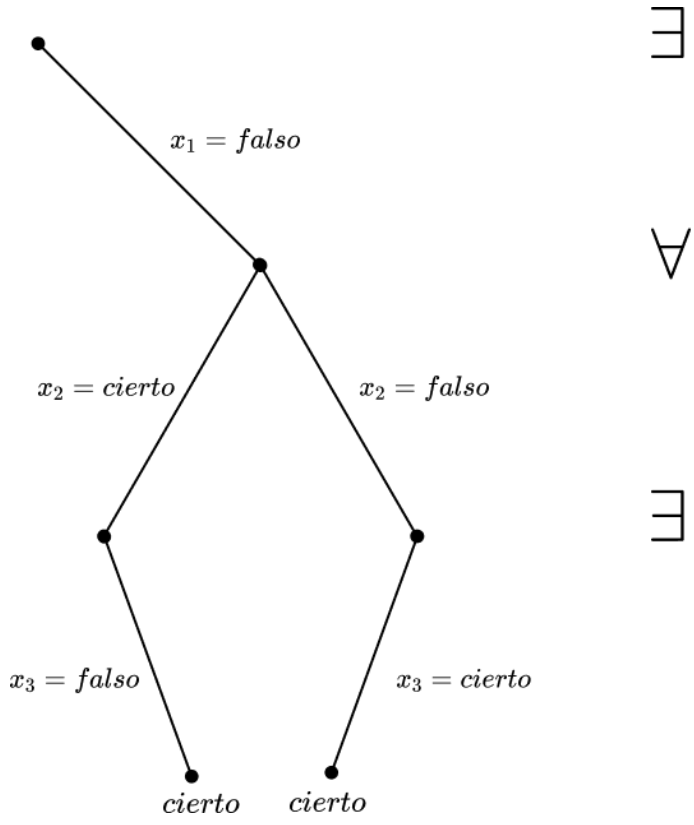

Figura 2.3: Árbol podado

podado de la figura 2.2, pero se puede extraer la misma información de ambos: hay una estrategia ganadora para el jugador ∃.

Para que el árbol podado de la figura 2.3 muestre que la QBF se cumple, es necesario

que ambas hojas sean ciertas. Una de ellas es el resultado de $\phi(\bot, \top, \bot)$ y la otra de $\phi(\bot, \bot, \top)$. Muchas fórmulas booleanas en tres variables cumplen esto; por ejemplo, lo cumple la fórmula $\Phi(x_1, x_2, x_3) = (x_1 \vee \neg x_1) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3)$, por lo que el árbol demuestra que la QBF $\exists x_1 \forall x_2 \exists x_3 \Phi(x_1, x_2, x_3)$ es cierta.

Este árbol puede ser válido para otras fórmulas booleanas de tres variables que tomen el lugar de ϕ. Sin embargo, solo es válido para QBF con los cuantificadores $\exists\forall\exists$ en ese orden, ya que están podados los niveles con un $\exists$ y no podado el nivel con el $\forall$.

Tras realizar el procedimiento arriba descrito, los vértices en un nivel con $\exists$ que son ciertos tienen ahora solo un hijo que es cierto. Adicionalmente los vértices en un nivel con un $\forall$ que sean ciertos también tienen solo hijos ciertos, por lo que todos los vértices ciertos tienen hijos ciertos que a su vez tienen más hijos ciertos. A su vez si todos los hijos de un vértice son ciertos este lo es también. Como esto se puede aplicar para todos los niveles podemos concluir que el vértice raíz es cierto si y solo si lo son todas las hojas del árbol podado.

No solo el árbol podado muestra la veracidad de una QBF, sino que además dada una QBF cierta se puede construir un árbol podado válido. Por tanto, si tenemos una QBF cierta sabemos que existe un árbol válido cuyas hojas son todas ciertas.

2.2.2. Reducción

En esta sección describiremos la reducción polinómica de TQBF (FNC) a MnOmO que utilizamos para demostrar la PSPACE-dureza de MnOmO. Para ello mostraremos los distintos elementos que la componen, explicando su propósito.

Viendo TQBF como un juego, el jugador $\exists$ ha de conseguir escoger los valores de las variables para que ϕ valga $\top$. Si aprovechamos que ϕ está en FNC, podemos descomponer este objetivo en hacer ciertas cada una de las cláusulas disyuntivas. Para poder imitar este objetivo en MnOmO, tendremos un objeto representando cada cláusula y haremos que un objeto llegando a su destino sea el equivalente a una cláusula de ϕ siendo cierta. Llamaremos al número de cláusulas y de objetos n. Llamaremos a las cláusulas $C_1, C_2, \ldots, C_n$.

Cada objeto necesita un vértice origen y un destino, llamaremos O a los orígenes y D a los destinos. El objeto i tendrá que ir de O_i a D_i, lo que nos da los valores de las funciones $o, d : \{1, 2, \ldots, n\} \to V$.

$$\begin{aligned} O_i, D_i \in V \quad & \forall i \in \{1, 2, \ldots, n\} \\ o(i) = O_i \quad & \forall i \in \{1, 2, \ldots, n\} \\ d(i) = D_i \quad & \forall i \in \{1, 2, \ldots, n\} \end{aligned}$$

Para que una cláusula disyuntiva de ϕ se cumpla, tiene que valer $\top$ al menos uno de sus literales. Podemos imitar esto en MnOmO ofreciendo a cada objeto i un camino

de O_i a D_i por cada literal distinto en C_i, y que ir por uno de ellos sea el equivalente a que se cumpla dicho literal.

En ϕ, si una cláusula se cumple porque x_i es cierto, no puede cumplirse otra porque $\neg x_i$ sea cierto y viceversa. Para imitar esto hace falta que si un objeto recorre su camino correspondiente a cumplir x_i, ningún otro pueda recorrer un camino correspondiente a cumplir $\neg x_i$. Para esto hace falta que ambos caminos utilicen la misma arista en sentido contrario, y que el tiempo límite no permita a unos objetos esperar hasta que pasen los otros para evitar una colisión. Por tanto, necesitamos una arista por cada variable, y todos los caminos de un origen a un destino tendrán que pasar por una de las aristas de cada variable.

Llamaremos P_i y N_i a los vértices de la arista correspondiente a la variable x_i. Los nombres de los vértices indican el sentido $P_i - N_i$ es "positivo" y recorrerlo equivale a que se cumpla x_i. Por el contrario recorrer el sentido "negativo" $N_i - P_i$ equivale a que se cumpla $\neg x_i$.

$$\begin{aligned} P_i, N_i \in V \quad & \forall i \in \{1, 2, \ldots, k\} \\ (P_i, N_i, 1) \in A \quad & \forall i \in \{1, 2, \ldots, k\} \end{aligned}$$

Donde k es el número de variables en la QBF.

En TQBF, no podemos escoger qué valor asigna a las variables el jugador $\forall$. Cuando asigna a una variable x_i el valor $\top$, ninguna cláusula puede cumplirse mediante el literal $\neg x_i$, y si le asigna $\bot$ no se podrá cumplir el literal x_i. Esto lo podemos imitar con un obstáculo que pueda bloquear uno de los dos sentidos de la arista $P_i - N_i$. Por tanto habrá un número de obstáculos igual al número de variables cuantificadas por un $\forall$. Llamaremos m a este número y M al conjunto de índices de estas variables. Cada obstáculo tendrá también un origen, distinto de los vértices que ya hemos introducido para que no pueda bloquear otras aristas relevantes. Llamamos a sus orígenes Om. Esto nos determina también el valor de la función $o_m : \{1, 2, \ldots, m\} \to V$.

$$M = \{i \in \{1, 2, \ldots, k\} : Q_i = \forall\} \tag{2.9}$$

$$\begin{aligned} Om_i \in V \quad & \forall i \in M \\ o_m(i) = Om_{M(i)} \quad & \forall i \in \{1, 2, \ldots, m\} \end{aligned}$$

Donde $M(i)$ es el i-ésimo elemento de M ordenado de menor a mayor.

El obstáculo correspondiente a la variable x_i tiene que poder desplazarse a los vértices P_i y N_i. De esta manera, si va al vértice P_i puede tomar la arista en el sentido $P_i - N_i$ bloqueando el sentido "negativo", y análogamente si va al vértice N_i puede bloquear el sentido "positivo". Así que si va hacia P_i es el equivalente a que el jugador $\forall$ asigne $x_i = \top$ y si va a N_i es el equivalente a que asigne $x_i = \bot$.

Para imitar el hecho de que estas asignaciones se hacen en el orden que están cuantificadas las variables, haremos que el obstáculo correspondiente a la variable x_i no tenga que tomar una decisión hasta que hayan pasado i unidades de tiempo. Así

conseguiremos que la variable x_1 se asigne en el momento 1, la variable x_4 en el momento 4, etc. Con este propósito, necesitaremos para cada obstáculo un vértice intersección Im donde puedan tomar la decisión. Según el índice, esta intersección estará más cerca o lejos del punto Om, para que la decisión se tome en el momento adecuado. La distancia de la intersección a los puntos P y N también varía, para que la distancia total de Om a P y N sea igual en todos los obstáculos.

$$\begin{aligned} Im_i \in V \quad & \forall i \in M \\ (Om_i, Im_i, i) \in A \quad & \forall i \in M \\ (Im_i, P_i, 2+k-i) \in A \quad & \forall i \in M \\ (Im_i, N_i, 2+k-i) \in A \quad & \forall i \in M \end{aligned}$$

De esta manera, el obstáculo i decide en el momento i si se dirige a P_i o a N_i, y llega a uno de estos dos vértices en el momento $2+k$. Si hacemos que los objetos también puedan alcanzar los vértices en ese mismo momento, conseguiremos que, si el obstáculo va a P_i y un objeto a N_i, o viceversa, pueda haber una colisión en la arista $P_i - N_i$. También conseguimos que, si el obstáculo y un objeto llegan al mismo vértice, P_i o N_i a la vez, no pueda haber una colisión, ya que cualquier arista la tomarían en el mismo sentido. Adicionalmente, al tener la arista $P_i - N_i$ longitud 1, si el objeto se detiene en su origen o en Im_i cualquier cantidad de tiempo (entera), entonces no llegará a tiempo para causar una colisión.

Por último, tenemos que completar los caminos que han de seguir los objetos. Si en la cláusula disyuntiva i aparece una variable x_i sin negar, queremos que el objeto i pueda ir desde O_i a P_i y desde N_i a D_i. Si aparece una variable negada, queremos que O_i conecte con N_i y P_i con D_i. Como en una cláusula puede haber varias variables, el objeto tendrá varios caminos disponibles y en algún momento el objeto deberá escoger cuál tomar.

Recordamos que las variables cuantificadas por un $\exists$ no tienen ningún obstáculo asociado que pueda bloquear la arista $P_i - N_i$ en ningún sentido, pero que si un objeto va a tomar uno de los dos sentidos sí bloquea el sentido contrario. Por tanto, si x_i está acompañada por un $\exists$, cuando mandamos a un objeto a P_i es el equivalente a que el jugador $\exists$ asigne la variable x_i a $\top$. Análogamente, mandar un objeto a N_i es el equivalente a asignarle $\bot$. Para imitar a TQBF, estas asignaciones las tenemos que hacer en el orden que están cuantificadas las variables, así que cada objeto i tendrá una intersección para cada variable que aparezca en la cláusula C_i y esté cuantificada con un $\exists$.

Dada una cláusula disyuntiva C_i, llamamos E_i al conjunto de los índices de variables cuantificadas con un $\exists$ que aparecen en un literal de C_i.

$$E_i = \{j : Q_j = \exists, x_j \in C_i\} \cup \{j : Q_j = \exists, \neg x_j \in C_i\} \tag{2.10}$$

Dado un índice $j \in E_i$ queremos que $s_i(j)$ sea el siguiente índice por orden numérico en E_i.

$$s_i(j) = min\{l : l \in E_i, l > j\} \tag{2.11}$$

Necesitamos finalmente crear los vértices intersección y conectar cada uno con el siguiente y con la arista de variable correspondiente. De esta manera, si el objeto decide no tomar el camino correspondiente a una variable, seguirá hacia la próxima intersección donde tendrá otra vez una elección.

$$\begin{aligned}
I_{i,j} \in V &\quad \forall i \in \{1,2,\dots,n\}, \forall j \in E_i \\
(O_i, I_{i,j}, j) \in A &\quad \forall i \in \{1,2,\dots,n\}, j = min(E_i) \\
(I_{i,j}, I_{i,s(j)}, s(j) - j) \in A &\quad \forall i \in \{1,2,\dots,n\}, \forall j \in E_i \backslash max(E_i) \\
(I_{i,j}, P_j, 2 + k - j) \in A &\quad \forall i \in \{1,2,\dots,n\}, \forall j \in E_i \text{ t.q. } x_j \in C_i \\
(I_{i,j}, N_j, 2 + k - j) \in A &\quad \forall i \in \{1,2,\dots,n\}, \forall j \in E_i \text{ t.q. } \neg x_j \in C_i
\end{aligned}$$

En cada vértice $I_{i,j}$, el objeto i decidirá si va a seguir el camino que lleva a D_i a través de la arista $P_j - N_j$ en alguno de sus dos sentidos, o si va a seguir hacia el siguiente vértice intersección. La primera elección equivale a que la cláusula i se cumpla gracias al literal x_j o $\neg x_j$, mientras que la segunda equivale a que se cumpla por algún otro literal cuya variable está cuantificada después. Adicionalmente, la primera elección equivale a asignarle a x_j $\top$ o $\bot$, ya que impide que otros objetos tomen el sentido contrario de la arista $P_j - N_j$, mientras que la segunda no equivale a darle ningún valor a x_j.

Aún tenemos que crear los caminos correspondientes a las variables cuantificadas por un $\forall$. Estas tienen un obstáculo asignado que en algún momento puede bloquear un sentido de la arista $P - N$. Si un objeto quiere usar dicha arista como parte de una estrategia que se asegura que no haya colisiones, solo puede dirigirse hacia P o N cuando el obstáculo ya ha bloqueado el sentido contrario. Por tanto la elección de ir a la arista de la variable x_i ha de tomarse después del momento i, ya que es a partir de este momento que se sabe qué sentido va a bloquear el obstáculo. Escogemos que esta elección se tome en la última intersección ya creada, a menos que se llegue a esta demasiado pronto, en cuyo caso se creará otra intersección "final" If más adelante.

$$\begin{aligned}
&\forall i \in \{1,2,\dots,n\} : \\
&\text{Si } \exists x_j \in C_i \text{ ó } \exists \neg x_j \in C_i \text{ t.q. } j > max(E_i) \\
&\text{Entonces } If_i \in V \text{ y } (I_{i,l}, If_i, k - l) \in A, \text{ donde } l = max(E_i)
\end{aligned} \tag{2.12}$$

$$\begin{aligned}
(I_{i,j}, P_l, 2 + k - j) \in A &\quad \forall i \in \{1,2,\dots,n\}, j = max(E_i), \forall l \in M \text{ t.q. } x_l \in C_i, l < j \\
(I_{i,j}, N_l, 2 + k - j) \in A &\quad \forall i \in \{1,2,\dots,n\}, j = max(E_i), \forall l \in M \text{ t.q. } \neg x_l \in C_i, l < j \\
(If_i, P_l, 1) \in A &\quad \forall i \in \{1,2,\dots,n\}, \forall l \in M \text{ t.q. } x_l \in C_i, l > max(E_i) \\
(If_i, N_l, 1) \in A &\quad \forall i \in \{1,2,\dots,n\}, \forall l \in M \text{ t.q. } \neg x_l \in C_i, l > max(E_i)
\end{aligned}$$

Que el objeto i tome una arista que le lleva a P_j o N_j con $j \in M$ equivale a que la cláusula i se cumpla gracias a que es cierto el literal x_j o $\neg x_j$. Sin embargo,

no es una elección que puedan tomar libremente los objetos o el controlador de los objetos. Esto se debe a que, si un obstáculo se está dirigiendo ya a N_j, ningún objeto podrá dirigirse a P_j, o de lo contrario no tendrá forma de evitar una colisión con el obstáculo en la arista $P_j - N_j$, y análogamente si un obstáculo se dirige a P_j, no podrán los objetos dirigirse a N_j. Esto hace que las decisiones que toman el objeto i en estos últimos vértices no equivalen a asignarle un valor a una variable, sino que equivale a ver, de entre las variables asignadas por un $\forall$, cuáles han sido asignadas de forma que se cumpla la cláusula.

Para terminar, tenemos que conectar todos los caminos de cada objeto a su destino.

$$\begin{aligned} (P_j, D_i, 1) \in A \quad & \forall i \in \{1, 2, \ldots, n\}, \forall j \text{ t.q. } \neg x_j \in C_i \\ (N_j, D_i, 1) \in A \quad & \forall i \in \{1, 2, \ldots, n\}, \forall j \text{ t.q. } x_j \in C_i \end{aligned}$$

Ya hemos descrito el grafo de la reducción. Solo falta especificar el tiempo límite. Este va a ser $t = k + 4$, que es la distancia del camino más corto entre el origen y el destino de un objeto. Esto hace que si los objetos toman caminos alternativos o si un obstáculo les retrasa, no puedan llegar a tiempo.

A continuación resumimos todos los vértices de la reducción:

$$\begin{aligned} O_i, D_i \quad & \forall i \in \{1, 2, \ldots, n\} \\ P_i, N_i \quad & \forall i \in \{1, 2, \ldots, k\} \\ Om_i, Im_i \quad & \forall i \in M \\ I_{i,j} \quad & \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \\ If_i \quad & \text{si } \exists x_j \in C_i \text{ ó } \exists \neg x_j \in C_i \text{ t.q. } j > max(E_i) \end{aligned} \tag{2.13}$$

Y todas las aristas:

$$
\begin{aligned}
(P_i, N_i, 1) &\quad \forall i \in \{1, 2, \ldots, k\} \\
(Om_i, Im_i, i) &\quad \forall i \in M \\
(Im_i, P_i, 2 + k - i) &\quad \forall i \in M \\
(Im_i, N_i, 2 + k - i) &\quad \forall i \in M \\
(O_i, I_{i,j}, j) &\quad \forall i \in \{1, 2, \ldots, n\}, j = min(E_i) \\
(I_{i,j}, I_{i,s(j)}, s(j) - j) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \backslash max(E_i) \\
(I_{i,j}, P_j, 2 + k - j) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \text{ t.q. } x_j \in C_i \\
(I_{i,j}, N_j, 2 + k - j) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \text{ t.q. } \neg x_j \in C_i \\
(I_{i,j}, P_l, 2 + k - j) &\quad \forall i \in \{1, 2, \ldots, n\}, j = max(E_i), \forall l \in M \text{ t.q. } x_l \in C_i, l < j \\
(I_{i,j}, N_l, 2 + k - j) &\quad \forall i \in \{1, 2, \ldots, n\}, j = max(E_i), \forall l \in M \text{ t.q. } \neg x_l \in C_i, l < j \\
(N_j, D_i, 1) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \text{ t.q. } x_j \in C_i \\
(P_j, D_i, 1) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \text{ t.q. } \neg x_j \in C_i \\
(I_{i,l}, If_i, k - l) &\quad \text{si } \exists x_j \in C_i \text{ ó } \exists \neg x_j \in C_i \text{ t.q. } j > l \text{ donde } l = max(E_i) \\
(If_i, P_l, 1) \in A &\quad \forall i \in \{1, 2, \ldots, n\}, \forall l \in M \text{ t.q. } x_l \in C_i, l > max(E_i) \\
(If_i, N_l, 1) \in A &\quad \forall i \in \{1, 2, \ldots, n\}, \forall l \in M \text{ t.q. } \neg x_l \in C_i, l > max(E_i)
\end{aligned}
\tag{2.14}
$$

donde n es el número de cláusulas, k el número de variables, el conjunto M está definido en 2.9, los conjuntos E_i en 2.10, las funciones s_i en 2.11 y m es el número de elementos de M.

En la próxima sección demostraremos que la instancia de TQBF tiene una respuesta afirmativa si y solo si también la tiene la instancia de MnOmO resultante de aplicar la reducción descrita en esta sección. Antes de ello, mostramos un ejemplo para facilitar la comprensión:

Instancia de TQBF

$$\forall x_1 \exists x_2 \exists x_3 (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \tag{2.15}$$

En esta instancia, el número de cláusulas es 2, por lo que para el problema MnOmO tomamos un número de objetos $n = 2$. Al haber 1 variable acompañada de un $\forall$, hemos de utilizar $m = 1$ obstáculo. Algunos conjuntos necesarios para crear el grafo son los siguientes: $M = \{1\}$, $E_1 = \{2\}$, $E_2 = \{2, 3\}$. Las funciones o, d y o_m vienen dadas por $o(1) = O_1$, $o(2) = O_2$, $d(1) = D_1$, $d(2) = D_2$ y $o_m(1) = Om_1$. Por último, el tiempo límite es $t = k + 4 = 7$ y el grafo es el representado en la figura 2.4.

El objeto 1 corresponde a la cláusula $x_1 \vee x_2$, por tanto tiene un camino por la arista $P_1 - N_1$ y $P_2 - N_2$, ambos en la dirección positiva ya que no aparecen negadas las variables. Como x_2 está cuantificado por un $\exists$ y aparece en segunda posición, la decisión de si ir al vértice P_2 se toma en el momento 2 cuando el objeto llega al vértice $I_{1,2}$. La variable x_1 está cuantificada por un $\forall$ y aparece en primera posición, por lo que el objeto no puede tomar la decisión de si dirigirse a P_1 hasta después del momento 1. Como el último (y único) vértice intersección es el $I_{1,2}$, al que se

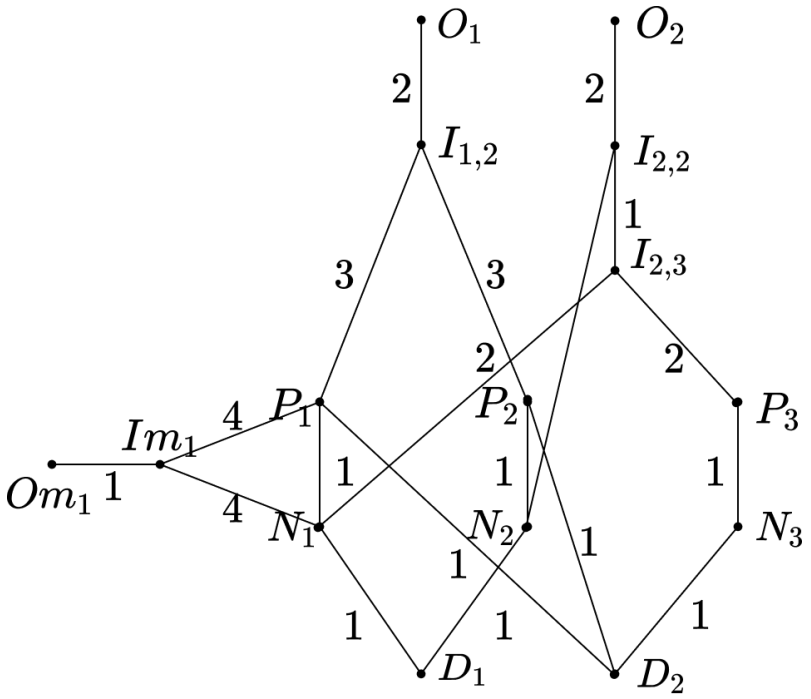

Figura 2.4: Grafo de la reducción

llega en el momento 2, no hace falta añadir otro vértice para tomar esta decisión y se puede hacer en $I_{1,2}$. Estos caminos se pueden ver con más claridad en el subgrafo de la figura 2.5.

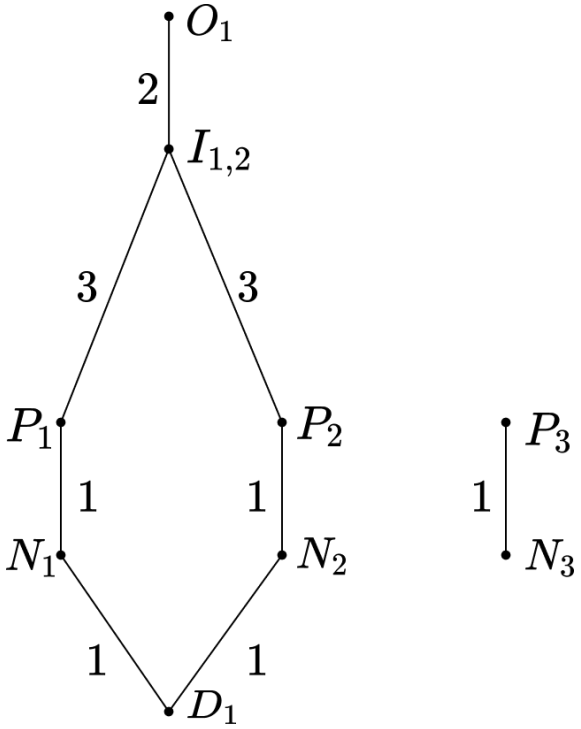

Figura 2.5: Subgrafo correspondiente al objeto 1

El objeto 2 corresponde a la cláusula $(\neg x_1 \vee \neg x_2 \vee x_3)$. Tiene por tanto un camino

por cada una de las aristas correspondientes a las tres variables. En el caso de las aristas $P_1 - N_1$ y $P_2 - N_2$ las recorre en el sentido negativo, que es el contrario al sentido que utiliza el objeto 1, por lo que no pueden utilizar ambos objetos la misma arista. La decisión de ir a N_2 se toma en el momento 2 y la de ir a P_3 en el momento 3, porque esas son las posiciones en las que aparecen cuantificadas con un $\exists$ las variables x_2 y x_3 respectivamente. La decisión de ir a N_1 se toma en el vértice $I_{2,3}$, ya que es el último vértice intersección y no es necesario añadir un vértice If. Estos caminos se pueden ver con más claridad en el subgrafo de la figura 2.6.

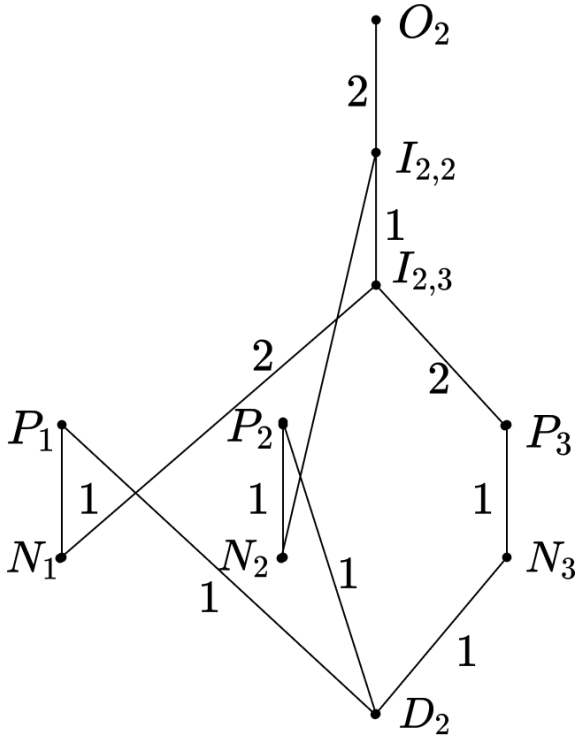

Figura 2.6: Subgrafo correspondiente al objeto 2

Al estar cuantificada con un $\forall$ la variable x_1, es necesario un obstáculo que haga el papel del jugador $\forall$ en TQBF. Este obstáculo sale del vértice Om_1 y cuando llega al vértice Im_1 en el momento 1 ha de tomar la decisión si se dirige a P_1 o a N_1. Si se dirige a P_1, el objeto 2 no podrá seguir el recorrido $O_2 - \cdots - N_1 - P_1 - D_1$, ya que existe el riesgo de que el obstáculo tome la arista $P_1 - N_1$ a la vez que el objeto 2 la toma en sentido contrario, causando una colisión. Sin embargo, en este mismo caso el objeto 1 sí puede utilizar el camino $O_1 - \cdots - P_1 - N_1 - D_1$, ya que alcanzará el vértice P_1 en el mismo momento que el obstáculo (momento 5) y no habrá riesgo de colisión.

Lema 2.2.1. *Si una instancia de TQBF tiene respuesta afirmativa, entonces su reducción a MnOmO también.*

Demostración. Para demostrar el lema 2.2.1, describiremos qué estrategia debemos seguir al mover los n objetos para que lleguen a su destino.

Primero de todo queremos observar que, según se ha construido el grafo, hay al menos un camino de O_i a D_i de longitud $k+4$. En concreto hay un camino de esta longitud por cada literal que aparezca en la cláusula i. Estos caminos tienen la forma $O_i - \cdots - I_{i,j} - P_l - N_l - D_i$ o $O_i - \cdots - I_{i,j} - N_l - P_l - D_i$,[1] donde entre O_i y $I_{i,j}$ pueden aparecer varios vértices $I_{i,s}$, y $j = l$ si x_j está acompañada de un $\exists$. Como $k+4$ es exactamente el tiempo límite, la estrategia no puede hacer que los objetos sigan otros recorridos ni que se detengan. De hecho, la estrategia se tiene que limitar simplemente a decidir, dinámicamente, qué camino de estos tomar. Si seguimos esta regla no tenemos que preocuparnos en absoluto del tiempo límite, sino de evitar colisiones.

La instancia de TQBF consta de un QBF en FNC que ha de ser cierta. Por tanto sabemos que existe para él un árbol podado válido cuyas hojas son todas ciertas. Nuestra estrategia utilizará este árbol para decidir a dónde debe dirigirse cada objeto.

Mientras se mueven los objetos por el grafo de la reducción, nos imaginaremos que movemos una "chincheta de decisión" por el árbol de juego. Esta chincheta comenzará en la raíz y la haremos descender nivel a nivel cada momento de la simulación hasta que alcance una hoja.

Diremos que una arista "encaja" si va hacia P_i y la chincheta está en un vértice del árbol de juego donde se le haya asignado $\top$ a x_i. Análogamente, también diremos que una arista "encaja" si va hacia N_i y la chincheta está en un vértice del árbol de juego donde se le haya asignado $\bot$ a x_i. No dejaremos ir a los objetos por aristas que no "encajen".

Los objetos comenzarán moviéndose todos hacia el vértice I más cercano. Cada obstáculo se quedará quieto o se moverá hacia el vértice Im_i, ya que es el único vértice conectado a su origen. Si un obstáculo se queda quieto, aunque sea solo 1 momento, no llegará a tiempo para bloquear a ningún objeto, así que podremos ignorarlo más adelante.

Los vértices $I_{i,j}$ están a distancia j del vértice O_i. Los vértices Im_j están a distancia j del vértice O_j. Además, si existe un vértice Im_j, entonces, no existe ningún $I_{i,j}$ para ningún i, ya que j ha de estar en M que no tiene intersección con ningún E_i (ver 2.13, 2.9, 2.10).

Por tanto, ignorando obstáculos que se hayan quedado parados, en un momento $j \in \{1, 2, \ldots, n\}$ puede ocurrir que un obstáculo llegue al vértice Im_j o que uno o más objetos lleguen a un vértice $I_{i,j}$ (cada objeto i a su $I_{i,j}$ distinto).

En el momento $j \in \{1, 2, \ldots, n\}$, la chincheta se encuentra en el nivel j del árbol. Si se da el primer caso arriba descrito, es porque $j \in M$ y por tanto x_j está cuantificada por un $\forall$ y el vértice donde se encuentra la chincheta tiene dos hijos. Esperaremos al siguiente momento y avanzaremos la chincheta según por donde siga el obstáculo j. Si toma la arista $Im_j - P_j$, la movemos al hijo correspondiente a asignarle $\top$ a la variable x_j. Si toma la arista $Im_j - N_j$, la movemos al otro hijo, correspondiente

[1] En algunos casos, If_i está en el lugar de $I_{i,j}$.

a asignarle $\bot$. Si el obstáculo se ha detenido o ha vuelto por la arista $Im_j - Om_j$, o incluso se había detenido previamente en Om_j, entonces movemos la chincheta a cualquiera de los dos hijos, ya que el obstáculo no va a llegar a tiempo a colisionar con ningún objeto.

Si se da el segundo caso, entonces x_j está cuantificada por un $\exists$ y nos encontramos en un vértice que solo tiene un hijo. En este caso movemos la chincheta a dicho hijo. Los objetos que se encuentren en un vértice $I_{i,j}$ tendrán una arista hacia P_j o N_j. A aquellos objetos cuya arista "encaje", según si este hijo corresponde a asignar $\top$ o $\bot$ a x_j, les haremos tomar la arista.

Adicionalmente, los objetos que no tomen las aristas $I_{i,j} - P_j$ o $I_{i,j} - N_j$ según el caso, han de seguir hacia el vértice $I_{i,s(j)}$, a menos que $j = max(E_i)$. En este caso, que sucede cuando el objeto está en su último vértice $I_{i,j}$, puede haber más aristas que pueda tomar hacia vértices P o N, correspondientes a variables condicionadas por un $\forall$.

Si una o más de estas aristas "encajan", según la posición de la chincheta, entonces el objeto debe continuar por cualquiera de ellas. Si por el contrario ninguna lo hace, entonces deberá continuar por la arista $I_{i,j} - If_i$. Como se ve en 2.12, no siempre existe dicha arista. Sin embargo, no puede ocurrir que no exista If_i y el objeto no haya tomado antes alguna arista anterior que "encajara", por razones que explicaremos algo más adelante.

Una vez el objeto esté en If_i, se repite el mismo procedimiento: el objeto puede dirigirse hacia cualquier arista que "encaje" según la posición de la chincheta. Siempre que un objeto haya llegado hasta If_i, "encajará" al menos una de estas aristas.

Ya hemos definido la estrategia. Para que esta estrategia demuestre el lema necesitamos que se cumplan las siguientes afirmaciones siempre que se siga la estrategia:

(i) A todo objeto le dejamos ir por algún camino de longitud $k + 4$.

(ii) No hay colisiones entre objetos.

(iii) No hay colisiones entre un objeto y un obstáculo.

(i) se cumple gracias a que en el árbol podado todas las hojas equivalen a una fórmula en FNC cierta. Esto implica que todas las cláusulas disyuntivas tienen algún literal cierto. Dada la cláusula C_i, se cumple al menos un literal suyo, que será x_j o $\neg x_j$ para algún j.

Si se trata de x_j, el objeto i tiene un camino que toma la arista $P_j - N_j$. Si x_j está cuantificada con un $\exists$, en el momento j el objeto i llegará al vértice $I_{i,j}$[2]. En este momento la chincheta estará en un vértice del nivel j y se moverá al hijo correspondiente a asignar $\top$ a x_j (de lo contrario no se cumpliría el literal x_j), por lo que la arista $I_{i,j} - P_j$ "encajará" y el objeto tendrá que tomarla.

Si por el contrario x_j está cuantificada por un $\forall$, entonces la chincheta está en un vértice que le asigna $\top$ a x_j desde el momento $j+1$, ya que es este momento cuando

[2] A menos que haya tomado otro camino, también de longitud $k + 4$, previamente.

se sabe a dónde se dirige el obstáculo j. El objeto llegará al vértice $I_{i,max(E_i)}$ o If_i en un momento posterior a j, por lo que ya estará asignado el valor a la variable y la arista $I_{i,max(E_i)} - P_j$ o $If_i - P_j$ "encajarán", por lo que el objeto podrá ir por ellas o por otra arista que "encaje" para seguir un camino de longitud $k + 4$.

En el caso en el que el literal que se cumple es $\neg x_j$ para alguna j, podemos seguir el mismo razonamiento con la diferencia de que el camino irá en el sentido $N_j - P_j$ y la chincheta estará en vértices que asignen $\bot$ a x_j.

Como todos los objetos están asociados a una cláusula, se les puede aplicar lo ya dicho, mostrando así que (i) es cierto.

(ii) se cumple porque las únicas aristas que tienen en común los caminos de distintos objetos son las aristas $P_j - N_j$, y solo dejamos a los objetos tomar aristas que "encajen". Si un objeto toma una arista que vaya a P_j sin haber pasado por N_j, se debe a que dicha arista "encaja", lo que por definición implica que la chincheta está en un vértice donde se le ha asignado $\top$ a x_j (por lo que no va a "encajar" ninguna arista que vaya a N_j sin pasar por P_j). Lo mismo sucede en el caso contrario: si dejamos que algún objeto vaya a N_j sin pasar por P_j, es porque la chincheta está en un vértice donde se ha asignado $\bot$ a x_j y no vamos a dejar que ningún objeto vaya a P_j sin pasar por N_j.

(iii) se cumple por razones similares a (ii). En este caso, no escogemos a dónde van los obstáculos, pero una vez un obstáculo se dirige a P_j, la chincheta se desplaza a un vértice donde $x_j = \top$, por lo que no vamos a permitir que ningún objeto se dirija a N_j sin pasar por P_j (no "encajaría" su arista). Pasa lo mismo en el caso contrario, cuando el obstáculo se dirige a N_j.

También es importante que, si un obstáculo se detiene, aunque sea durante 1 momento, no puede colisionar con ningún objeto. Esto se debe a que los objetos pasan por los vértices P_j y N_j en los momentos $2 + k$ y $3 + k$, y el obstáculo, si se ha detenido, llegará a uno de ellos como pronto en el momento $3 + k$. Aunque este sea el mismo vértice donde se encuentra un objeto, si el objeto toma la arista hacia su destino, entonces no hay posibilidad de que el obstáculo atraviese la misma arista en sentido contrario causando una colisión.

Lo mismo sucede si el obstáculo no se detiene y tanto el obstáculo como un objeto llegan al mismo vértice P_j o N_j en el momento $2 + k$: al estar en el mismo vértice, el obstáculo no puede causar una colisión.

También es aquí importante la regla 3 introducida en la definición del problema. Si a los obstáculos se les permitiera ir a los destinos de los objetos, podrían hacerlo en $k + 3$ momentos (tomando el camino más corto), mientras que los objetos tardan $k + 4$ momentos, por lo que podría haber una colisión en la última arista. Gracias a la regla 3, los únicos vértices en los que puede estar un obstáculo antes de que pase un objeto por ellos son los P y N, ya que los vértices I están más cerca de los objetos (sin contar los Im) y no pueden ir los obstáculos a los vértices D.

Viendo que se cumplen (i), (ii) y (iii) cuando la QBF es cierta, sabemos que la instancia de MnOmO creada por la reducción tiene una respuesta afirmativa siempre

que también la tenga la instancia de TQBF correspondiente. □

Lema 2.2.2. *Si una instancia de TQBF tiene respuesta negativa, entonces su reducción a MnOmO también.*

Demostración. Para demostrar el lema 2.2.2, demostraremos que si la reducción a MnOmO tiene una respuesta afirmativa, entonces la instancia de TQBF también. Esto lo haremos transformando una estrategia que resuelva la reducción en una estrategia ganadora para el jugador ∃ en el juego TQBF.

Recordamos que una estrategia en una instancia MnOmO ha de saber qué decisiones tomar para cualquier conjunto de movimientos de los obstáculos. En esta demostración, sabemos que existe una estrategia que garantiza la llegada de los objetos sin colisiones entre ellos ni con obstáculos. Para transformarla en una estrategia ganadora para el jugador ∃, solo necesitamos saber su comportamiento para algunas de las combinaciones de movimientos de los obstáculos.

Concretamente, vamos a utilizar el comportamiento de la estrategia cuando todos los obstáculos toman alguno de los dos caminos $Om_i - Im_i - P_i - N_i$ o $Om_i - Im_i - N_i - P_i$ sin detenerse. Elegimos estos caminos porque son equivalentes a que el jugador ∀ elija asignarle ⊤ o ⊥ a la variable x_i.

Cada obstáculo tiene dos opciones, por lo que hay 2^m combinaciones, donde m es el número de obstáculos. Utilizaremos lo que hace la estrategia de la reducción en cada una de estas combinaciones para construir un árbol de juego podado que represente una estrategia ganadora para el jugador ∃.

Para ello, recordamos que los vértices Im_i están a distancia i de Om_i, y al no detenerse, los obstáculos lo alcanzarán en el momento i. Similarmente, los vértices $I_{i,j}$ están a distancia j de los vértices O_i, en ocasiones atravesando varias aristas. Los caminos de O_i a D_i tienen distancia $4+k$, que es el tiempo límite, por lo que los objetos tampoco se podrán detener y llegarán a $I_{i,j}$ en el momento j. Adicionalmente, para cada variable x_i cuantificada por un ∀ existe exactamente un vértice Im_i, y para toda variable x_j cuantificada por un ∃ que aparezca en alguna cláusula existe al menos un vértice $I_{l,j}$. Asumiremos que toda variable cuantificada con un ∃ aparece en alguna cláusula, en caso contrario cualquier estrategia ganadora del jugador ∃ puede asignarle cualquier valor.

Diremos que un vértice de un árbol podado V es equivalente a un momento de una simulación si se cumplen las tres afirmaciones siguientes:

- Si x_i está cuantificada con un ∀ y le ha sido asignado ⊤ en V, el obstáculo i se encuentra en la arista $Im_i - P_i$.
- Si x_i está cuantificada con un ∀ y le ha sido asignado ⊥ en V, el obstáculo i se encuentra en la arista $Im_i - N_i$.
- Si x_i está cuantificada con un ∀ y aún no ha sido asignado ningún valor en V, el obstáculo i se encuentra en la arista $O_i - Im_i$.

Como las variables se asignan en el mismo orden en que los obstáculos pasan por sus vértices Im, todo vértice es equivalente a al menos un momento de alguna simulación posible. Los vértices donde es el turno del jugador $\exists$ son equivalentes a los mismos momentos de las mismas simulaciones que sus hijos, pero vamos a utilizar esta definición para distinguir vértices de distintas ramas pero mismo nivel por lo que esto no nos preocupa.

Por tanto, en cada uno de los momentos $j \in \{1, 2, \ldots, k\}$ de una simulación sucederá exactamente una de las dos siguientes cosas:

- Un obstáculo llegará al vértice Im_j.
- Uno o más objetos llegarán a sus vértices $I_{i,j}$, cada uno con un i diferente. Diremos que estos objetos son activos en el momento j.

En el primer caso los vértices del nivel j del árbol de decisión tendrán los dos hijos posibles, ya que corresponde a un turno del jugador $\forall$.

En el segundo caso, queremos que dichos vértices del árbol de decisión podado tengan sólo un hijo, por lo que tendremos que decidir cuál de los dos posibles va a ser. La estrategia de la reducción puede tomar distintas decisiones según lo que haya sucedido en los momentos anteriores, por lo que similarmente algunos vértices del nivel j podrán tener el hijo equivalente a asignarle $\top$ a x_j mientras otros tienen el hijo equivalente a asignarle $\bot$.

Si en una simulación dada la estrategia en la reducción envía alguno de los objetos activos al vértice P_j, entonces el vértice del nivel j equivalente a esta simulación en el momento j debe tener el hijo correspondiente a asignarle $\top$ a x_j. Análogamente, para aquellas simulaciones en las que un objeto activo se mande hacia el vértice N_j, su vértice del nivel j del árbol equivalente en el momento j ha de tener el hijo correspondiente a asignarle $\bot$ a x_j.

En las simulaciones en las que no suceda ni una cosa ni la otra, los vértices equivalentes a dichas simulaciones pueden tener cualquiera de los dos hijos, por ejemplo el que asigna a $\top$ si se quiere ser sistemático. En ninguna simulación ocurre que se envía algún objeto activo a P_j y algún otro a N_j, ya que para llegar a su destino tendrían que tomar la arista $P_j - N_j$ en sentidos contrarios simultáneamente, causando una colisión.

Siguiendo estas instrucciones comenzando en el momento 1 y llegando hasta el momento k, podemos construir todo el árbol de juego podado. Es necesario que tengamos en cuenta las 2^m distintas simulaciones posibles, o de lo contrario alguna rama quedaría incompleta.

Para demostrar que este árbol podado describe una estrategia ganadora, basta con ver que en todas sus hojas la fórmula es cierta tras hacer todas las asignaciones de valores a las variables. Aprovechamos que la fórmula está en FNC por lo que basta con ver que se cumplen todas las cláusulas disyuntivas, cada una representada por un objeto de la reducción.

Fijémonos en una hoja cualquiera, que es equivalente a una simulación en concre-

to a partir de su momento $k+1$, cuando ya todos los obstáculos han tomado su decisión. Además, si otro vértice del árbol es equivalente a la misma simulación en otro momento previo j, entonces esta hoja es descendiente de este vértice, ya que hasta la variable j tienen las mismas asignaciones de variables, y tiene sin asignar las variables posteriores cuantificadas por un $\forall$.

Podemos dividir a los objetos de la simulación equivalente a la hoja en cuatro categorías según qué camino están siguiendo. Según esta categoría se cumplirá su cláusula correspondiente por una razón u otra. Se tratan de las siguientes:

(i) El objeto i va por una arista $I_{i,j} - P_j$.

(ii) El objeto i va por una arista $I_{i,j} - N_j$.

(iii) El objeto i va por una arista $I_{i,l} - P_j$ con $l \neq j$ o por una arista $If_i - P_j$.

(iv) El objeto i va por una arista $I_{i,l} - N_j$ con $l \neq j$ o por una arista $If_i - N_j$.

En primer lugar, recordamos que la construcción del grafo implica que en los casos (i) y (iii) la cláusula i contiene el literal x_j, y que en los casos (ii) y (iv) contiene el literal $\neg x_j$. También es importante recalcar el hecho de que estos casos son exhaustivos: no existen más aristas que permitan llegar de O_i a D_i, y, como la estrategia lleva a todos los objetos a su destino, ha de hacerlo por alguna de esas aristas.

En los casos (i) y (ii), el objeto tomó dicha arista en el momento j de la simulación, cuando era un objeto activo. Por tanto, en el caso (i) esto causa que un vértice en el nivel j equivalente con ese momento de la simulación tenga como único hijo aquél donde se asigna $\top$ a x_j. Como la hoja en la que nos estamos fijando ha de ser también descendiente de este vértice, también tiene $\top$ asignado a x_j. Esto hace que la cláusula i sea cierta al contener el literal x_j. En el caso (ii) pasa algo similar solo que se le asigna $\bot$ a x_j, lo cual también hace cierta a la cláusula puesto que contiene el literal $\neg x_j$.

En los casos (iii) y (iv), el objeto ha tomado dicha arista en su último o penúltimo (si existe If_i) vértice intersección, ya que x_j en estos casos se trata de una variable cuantificada por un $\forall$ y no existe el vértice $I_{i,j}$. Esto implica que existe un obstáculo j que está dirigiéndose a P_j o N_j. En el caso (iii) no se puede estar dirigiendo a N_j, ya que llegaría a N_j en el mismo momento que el objeto llega a P_j, y como el objeto ha de continuar por la arista $P_j - N_j$ podría haber una colisión. Por ello, el obstáculo se dirige a P_j. Como la hoja es equivalente a esta simulación, esto significa que en la hoja x_j ha sido sustituida por $\top$ (ver la definición de equivalencia). Esto causa que la cláusula i se cumpla ya que contiene el literal x_j.

En el caso (iv), ocurre que el objeto ha de estar dirigiéndose a N_j y en la hoja se cumple $x_j = \bot$, por lo que la cláusula i también se cumple al contener $\neg x_j$.

Como hemos escogido una hoja cualquiera, esto ocurre en todas las hojas por lo que todas las hojas del árbol de juego podado son ciertas, lo que implica que representa una estrategia ganadora para el jugador $\exists$.

Podemos concluir que, si la reducción a MnOmO de una instancia TQBF tiene respuesta afirmativa, entonces la instancia original tambien la tenía. Esto es equiva-

lente a que si la instancia de TQBF tiene una respuesta negativa también la tiene su reducción a MnOmO

□

Teorema 2.2.1. *MnOmO es PSPACE-duro.*

Demostración. El teorema 2.2.1 es consecuencia de los lemas 2.2.2 y 2.2.1, de que la reducción descrita es polinómica y de que TQBF es PSPACE-duro.

La reducción es polinómica puesto que cuando la instancia de TQBF tiene n cláusulas y k variables, el número de vértices de la reducción está acotado por $2n+2k+nk$ y el número de aristas por $2n+2kn$. Estos son los números de vértices y aristas resultantes de aplicar una reducción a una instancia TQBF donde todas las variables están cuantificadas con un $\exists$ y todas las cláusulas son una disyunción de todas las variables. □

2.3. PSPACE-completitud

Que un problema sea PSPACE-completo significa que es PSPACE-duro y que pertenece a PSPACE. Sabiendo ya que es PSPACE-duro solo nos falta comprobar que pertenece a PSPACE.

PSPACE es la clase computacional compuesta por todos los problemas que pueden ser resueltos utilizando una cantidad polinómica de espacio. TQBF pertenece a PSCACE (Garey y Johnson, 1979). Podemos ver este hecho con ayuda del árbol de la figura 2.1. Este árbol representa todas las opciones posibles de asignar las variables.

Si comprobamos el valor de la fórmula en las hojas del árbol, podemos saber el valor en el nivel superior y así sucesivamente. El número de hojas es 2^k, por lo que el algoritmo "naive" que consiste en generar todo el árbol no utiliza una cantidad de espacio polinómica. Sin embargo, no es necesario generar todo el árbol a la vez. Se puede computar el valor de cada hoja individualmente, y simplemente recordar dónde nos encontramos en el árbol y seguir un orden preestablecido con el objetivo no repetir hojas. Por ejemplo, si se implementa como una función recursiva, cada vértice podría llamar primero a su hijo correspondiente a asignar $\top$ a la primera variable, después a su otro hijo, y finalmente devolver su propio valor según el valor de sus hijos. Una vez un vértice devuelve un valor, no importa qué valor tuvieran sus hijos, así que no hace falta guardar información de todo el árbol. En memoria habría que guardar simplemente las llamadas recursivas en proceso. Esto implica que el algoritmo es lineal, y por tanto polinómico, en espacio, puesto que la altura del árbol es el número de variables del problema.

Una implementación iterativa de este mismo algoritmo no necesita guardar un *stack* de llamadas. Sin embargo, sigue necesitando alguna manera de guardar en qué punto del árbol se encuentra la ejecución, para saber por dónde proseguir. Esta información depende de la altura del árbol, que en el caso de TQBF es lineal. Mediante este

razonamiento podemos concluir que si tenemos un problema que pueda resolverse probando muchas combinaciones, y estas combinaciones se pueden representar en un árbol similar al de TQBF y con altura polinómica, el problema está en PSPACE. Nos preguntamos por tanto si este razonamiento se puede utilizar para ver si MnOmO pertenece a PSPACE.

Razonemos qué estructura tiene el árbol que necesitamos para resolver MnOmO. Aunque en la demostración del lema 2.2.2 hayamos utilizado los árboles de TQBF para modelizar el movimiento de los objetos y obstáculos, esto en general no es posible en instancias de MnOmO. De hecho, ni siquiera en dicha demostración estábamos considerando todos los casos, sino solo algunos de los seguidos por una estrategia correcta. En un caso cualquiera con un grafo cualquiera no contamos con información sobre la forma de una estrategia correcta, por lo que no podemos tomar "atajos" basándonos en ella. Por tanto debemos tener en cuenta todas las posibles acciones de los objetos y obstáculos, desde detenerse indefinidamente, hasta avanzar y retroceder por la misma arista.

En el caso general, siempre que un objeto u obstáculo esté en un vértice, dicho objeto puede comenzar a desplazarse a cualquier otro vértice adyacente o mantenerse quieto. En el mismo momento de la simulación, puede haber muchos objetos y muchos obstáculos en algún vértice. Por tanto, si representamos un momento de la simulación con un vértice y tiene un hijo por cada combinación de movimientos diferente posible, puede tener una gran cantidad de hijos. Podemos acotar este número de hijos por $n * |V| * m * (|V| - n)$, donde $|V|$ es el número de vértices en el grafo. Llegamos a este número ya que cada uno de los n objetos tiene un máximo de $|V|$ opciones (quedarse quieto o dirigirse a un vértice distinto al donde está), y cada uno de los m obstáculos tiene a lo sumo $|V| - n$ opciones, ya que no puede ir a ninguno de los n destinos. Esta cota superior solo se alcanza en grafos muy densos, pero es posible.

Si construimos el grafo de esta manera, podemos ver las ramas pueden tener altura $t + 1$, donde t es el tiempo límite de la instancia. Solo tienen menor altura aquellas donde en algún momento todos los objetos y obstáculos se encontraban en una arista y por tanto no tenían ninguna decisión que tomar. Podemos reducir la altura de más ramas si evitamos hacer nuevos hijos cuando todos los objetos y obstáculos que tienen alguna decisión que tomar se quedan quietos. Para diferenciar entre que los objetos y obstáculos hagan "X" inmediatamente, o que hagan "X" tras esperar k momentos, podemos simplemente acompañar cada vértice con el momento de tiempo que representa. Sin embargo, después de hacer esto es posible que siga habiendo ramas de altura $t + 1$.

La altura de las ramas es un inconveniente. Como hemos mencionado, si fuese polinómica podríamos afirmar que el problema pertenece a PSPACE. Hay que aclarar que cuando decimos polinómica nos referimos a polinómica respecto con el tamaño utilizado para representar la instancia. El tiempo límite t solo es polinómico si se representa en base unaria. Si se representa con cualquier otra base $b \neq 1$, entonces con añadir un solo dígito 0 a la representación, t se ve multiplicado por b. Esto resulta en que t es exponencial, por lo que el árbol no tiene altura polinómica.

Podemos reflexionar si hay alguna forma de limitar el número de combinaciones de movimientos que consideramos. En un trabajo similar pero sin obstáculos, se aprovechaba el hecho de que no había colisión en los vértices para concluir que no era necesario que un objeto pasara más de una vez por el mismo vértice (Carreño López, 2024). Esto serviría para acotar polinómicamente el número de veces que se mueve un objeto en una misma rama, y si posteriormente acotamos los movimientos de los obstáculos podríamos llegar a ramas de longitud polinómica. Desafortunadamente, no es cierto que en MnOmO nunca sea necesario que un objeto pase dos veces por el mismo vértice, ya que puede necesitarlo para "esquivar" obstáculos. Se puede ver esto con la instancia contraejemplo descrita por 2.16 y la figura 2.7.

$$n = 1, \quad m = 1, \quad o(1) = A, \quad d(1) = D, \quad o_m(1) = B, \quad t = 6 \tag{2.16}$$

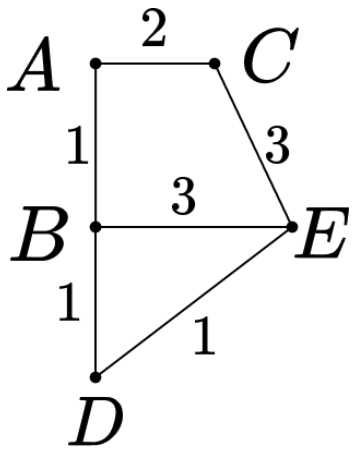

Figura 2.7: Grafo del contraejemplo

En esta instancia, el único objeto ha de ir de A a D evitando al obstáculo en B. Para ello no puede comenzar dirigiéndose a B, ya que existiría la posibilidad de que el obstáculo comience a moverse hacia A en el mismo momento causando una colisión. Por tanto, la estrategia correcta es dirigirse a C. Una vez ha llegado el objeto a C, puede seguir hacia E a menos que el obstáculo esté ya en la arista $B - E$. En ese caso, el objeto ha de volver a A y desde ahí ir a D a través de B, ya que el obstáculo no está en B y no puede volver a tiempo. Resumiendo, el objeto ha de seguir el camino $A - C - E - D$ o $A - C - A - B - D$ según lo que haga el obstáculo.

No podemos afirmar que los objetos no tengan que repetir vértices. pues de hecho en esta instancia contraejemplo, si el obstáculo toma la arista $B - E$, el objeto ha de pasar una segunda vez por A. Es posible que se pueda acotar el número de veces que un objeto puede necesitar pasar por el mismo vértice, por ejemplo con el número de obstáculos. Sin embargo, no hemos logrado hacerlo en este trabajo, por lo que queda como posible idea para en un trabajo futuro demostrar que MnOmO pertenece a PSPACE.

En caso de que esta idea no resultara posible, o no fuese suficiente por sí sola, podría utilizarse también otra manera de acotar el número de movimientos distintos que

pueden hacer los objetos. Esta manera se refiere a utilizar alguna heurística que pueda garantizarnos que en ciertas posiciones ya no es posible alcanzar el objetivo. Por ejemplo, en el grafo de la figura 2.7, si hubiera un obstáculo en B y otro en E, no sería posible que el objeto llegase sin riesgo de colisión, por lo que no haría falta continuar con esta rama. Este método requeriría acotar por separado aquellas ramas que sí lleven a los objetos a sus destinos, por lo que tendría que ser combinado con alguna otra idea como la anterior, o con una demostración de que si la respuesta a la instancia es afirmativa, la altura de las ramas que lo prueben es polinómica.

En definitiva, no hemos podido acotar el número de movimientos de los objetos, aunque hay posibles ideas de cómo podría ser hecho. Incluso si demostráramos que estos movimientos se pueden acotar polinómicamente, aún quedarían los movimientos de los obstáculos como inconveniente, ya que también se pueden mover un número exponencial de veces.

En primer lugar, hay que recalcar que es imposible afirmar que el número de movimientos hecho por un obstáculo está acotado polinómicamente. Con que haya una arista de longitud 1, ya solo puede acotarse por t, que crece exponencialmente. Pese a ello, es posible que pueda manipularse la construcción del árbol para evitar tener que considerar movimientos de los obstáculos que no afectan a los objetos, como por ejemplo si un obstáculo va y vuelve por la misma arista repetidas veces.

En cualquier caso podemos ver que, si MnOmO pertenece a PSPACE, esto es difícil de probar, debido a la cantidad exponencial de movimientos que pueden hacer tanto los objetos como los obstáculos. Por tanto podemos afirmar que es un problema PSPACE-duro pero no podemos afirmar que sea PSPACE-completo.

Si consideramos variantes de MnOmO donde el número de movimientos está acotado de cierta forma, entonces estas variables podrán pertenecer a PSPACE. Adicionalmente, si en estas variantes se sigue aplicando la reducción desde TQBF, entonces serán problemas PSPACE-completos. Estudiaremos variantes de esta forma, junto con otras variantes de MnOmO en el siguiente capítulo.

Capítulo 3

Variantes de MnOmO

En este capítulo discutimos las propiedades de varias variantes de MnOmO. Las primeras tratan de averiguar cómo podemos cambiar el problema ligeramente para que sea PSPACE-completo, mientras que las que siguen exploran qué sucede si modificamos detalles de la definición. Cuando decimos "variante de MnOmO" nos referimos a un problema con una definición muy similar a MnOmO, donde nos fijamos especialmente en las diferencias con MnOmO.

3.1. Variantes PSPACE-completas

Hemos visto en el capítulo anterior que aún desconocemos si MnOmO pertenece a PSPACE, por lo que no podemos afirmar que sea PSPACE-completo. Esto es consecuencia de que no hemos logrado acotar la cantidad de movimientos que pueden llegar a hacer los objetos y obstáculos. En esta sección mostramos varias maneras de acotar esta cantidad en la definición del problema para llegar a una variante PSPACE-completa.

3.1.1. MnOmO-k

Una de las posibles formas de acotar los movimientos de los objetos que mencionamos en el anterior capítulo consistía en que los objetos no necesitaran pasar varias veces por el mismo vértice. Vimos que no es cierto que no lo necesiten, pero sería conveniente, ya que implicaría que cada objeto no necesita desplazarse más veces que el número de vértices.

Las variantes MnOmO-k surgen de tomar esta idea de que los objetos no necesiten pasar varias veces por el mismo vértice y cambiarla por que los objetos no puedan hacerlo. Para ello definimos que, para todo k natural, MnOmO-k es una variante de MnOmO donde cada objeto y cada obstáculo no puede pasar más de k veces por el mismo vértice. Por ejemplo, MnOmO-1 no permite ni a los objetos ni a los obstáculos

pasar más de una vez por el mismo vértice. Tenemos que restringir también el movimiento de los obstáculos, ya que de lo contrario nos surgirían algunos de los problemas que surgen al intentar demostrar que MnOmO pertenece a PSPACE.

Por tanto, en las variantes MnOmO-k, el número de movimientos de cada objeto y obstáculo puede ser acotado por $k|V|$, donde $|V|$ es el número de vértices del grafo. Esto hace que las ramas del árbol que representa todas las combinaciones posibles puedan tener una altura de a lo sumo $n * m * k * |V|$, ya que podría ser que en cada momento comience un movimiento un único objeto u obstáculo. Por tanto estas ramas están acotadas polinómicamente, por lo que un algoritmo podría recorrerlas todas para resolver el problema utilizando una cantidad polinómica de memoria. Con esto podemos concluir que MnOmO-k pertenece a PSPACE para todo k natural.

Que las variantes MnOmO-k son PSPACE-duras es relativamente sencillo de ver, ya que en la reducción de TQBF a MnOmO no es necesario que ningún objeto pase varias veces por el mismo vértice. Esto hace que el lema 2.2.1, que afirma que si es afirmativa una instancia de TQBF también lo es la instancia de MnOmO, se siga cumpliendo al sustituir MnOmO por MnOmO-k, para todo k natural. Adicionalmente, que los obstáculos no puedan pasar varias veces por el mismo vértice no hace que más instancias creadas por la reducción sean ciertas, por lo que se sigue cumpliendo el lema 2.2.2, que afirma que si es negativa una instancia de TQBF también lo es la instancia de MnOmO, tras sustituir MnOmO por MnOmO-k, para todo k natural.

Debido a todo lo anterior, podemos afirmar que, para todo k natural, MnOmO-k es PSPCAE-completo.

3.1.2. MnOmO-p

Como nuestro objetivo es limitar el número de movimientos para que sea polinómico, introducimos las variantes MnOmO-p que lo hacen de forma directa. Dado un polinomio p, definimos MnOmO-p como una variante de MnOmO cuya diferencia es que ni los objetos ni los obstáculos pueden hacer más de $p(n)$ movimientos, donde n es el tamaño de la instancia.

Evidentemente el tamaño de la instancia se puede medir de varias maneras, pero independientemente de cómo lo hagamos el resultado será un problema perteneciente a PSPACE, por las mismas razones por las que las variantes MnOmO-k pertenecen a PSPACE.

Para que demostrar que una variante MnOmO-p es PSPACE-dura con la reducción de la sección 2.2, necesitamos que para cualquier instancia creada por la reducción de TQBF a MnOmO, se cumpla que todo camino de distancia mínima entre un origen de objeto y su destino tenga un número de aristas menor o igual a $p(n)$, donde n es el tamaño de la instancia. Como esta condición es algo abstracta y poco útil, podemos crear una algo más restrictiva. Para ello nos hará falta especificar una manera concreta de medir el tamaño de una instancia.

Por simplicidad, midamos el tamaño de la instancia como el número de vértices más el número de aristas. Las instancias resultantes de aplicar una reducción tienen un tamaño mínimo de $6n+5m+3k$, donde n, m y k son el número de objetos, obstáculos y variables (en la instancia TQBF) respectivamente. El número de aristas mínimo en un camino entre el origen de un objeto y su destino es de 4 aristas, como por ejemplo, un camino $O_i - I_{i,j} - P_j - N_j - D_i$. Esta cifra sigue siendo posible en las instancias más pequeñas, donde $n = 1$, $m = 0$, $k = 1$ (exigimos que la instancia de TQBF tenga al menos una variable y una cláusula), lo que hace que el tamaño de la instancia sea 9. Por tanto, si p cumple la siguiente condición 3.1, podemos afirmar que MnOmO-p es PSPACE-duro. Si queremos generalizar más, la propiedad 3.1 la cumplen todos los polinomios con coeficientes naturales y grado mayor o igual a 1, por lo que esta también es una condición suficiente.

$$p(x) \geq 4 \quad \forall x \geq 9 \tag{3.1}$$

Podemos afirmar por tanto, que dado un polinomio p que cumpla 3.1, la variante MnOmO-p es PSPACE-completa.

3.2. MnOmO-vértices

Una variante muy interesante es aquella que surge cuando extendemos el conjunto de reglas para que también pueda haber colisiones en los vértices. Llamamos a esta variante MnOmO-vértices. La nueva regla podría tener la siguiente forma:

9. Si dos objetos u obstáculos se encuentran en el mismo vértice en el mismo momento, se dice que hay una colisión entre ambos.

Esto implicaría que una estrategia correcta ha de evitar que los objetos se encuentren simultáneamente en un mismo vértice, o que un obstáculo pueda alcanzar un vértice donde se encuentra un objeto.

La reducción a partir de TQBF descrita en la sección 2.2.2 no sirve para demostrar la PSPACE-dureza de esta variante, ya que requiere que los objetos puedan coincidir entre sí y con los obstáculos en los vértices P y N. Para adaptarla tenemos que hacer varias cosas que describimos a continuación, que se limitan a cambiar el tiempo límite y la longitud de las aristas ya existentes, afortunadamente.

En primer lugar, si varias cláusulas se cumplen con el mismo literal, varios objetos van a tener que pasar por la misma arista. Si lo hiciesen a la vez, causarían una colisión en un vértice. Por ello, han de pasar en momentos separados. Con que cada uno espere un momento más que el anterior es suficiente para arreglar este problema. Si se cumplen todas las cláusulas por el mismo literal, el primer objeto habrá de esperar 0 momentos, el segundo 1, y el n-ésimo $n - 1$. En la reducción a MnOmO ningún objeto puede esperar, ya que el tiempo límite es igual a la longitud de los caminos más cortos. Por tanto, al adaptarla para que sea una reducción a MnOmO-

vértices, es necesario que el tiempo límite supere en $n-1$ a la longitud de los caminos que unen los orígenes y los destinos.

Aumentar el tiempo límite causa un problema distinto: con más tiempo límite los objetos pueden seguir caminos más largos y llegar a tiempo. Por ejemplo, si aplicamos la reducción del capítulo anterior a la instancia de TQBF con la QBF 3.2, llegamos a una instancia de MnOmO con el grafo de la figura 3.1 y tiempo límite $t = k + 4 = 7$. Si ahora aumentamos en $n - 1 = 1$ dicho tiempo límite, para que ambos objetos puedan utilizar la misma arista si lo necesitan, nos queda $t = 8$, y el objeto 1 podría seguir el caminos $O_1 - I_{1,2} - P_1 - D_2 - N_2 - D_1$ que tiene longitud 8 pero no es uno de los caminos que queremos permitir.

$$\forall x_1 \exists x_2 \exists x_3 (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \tag{3.2}$$

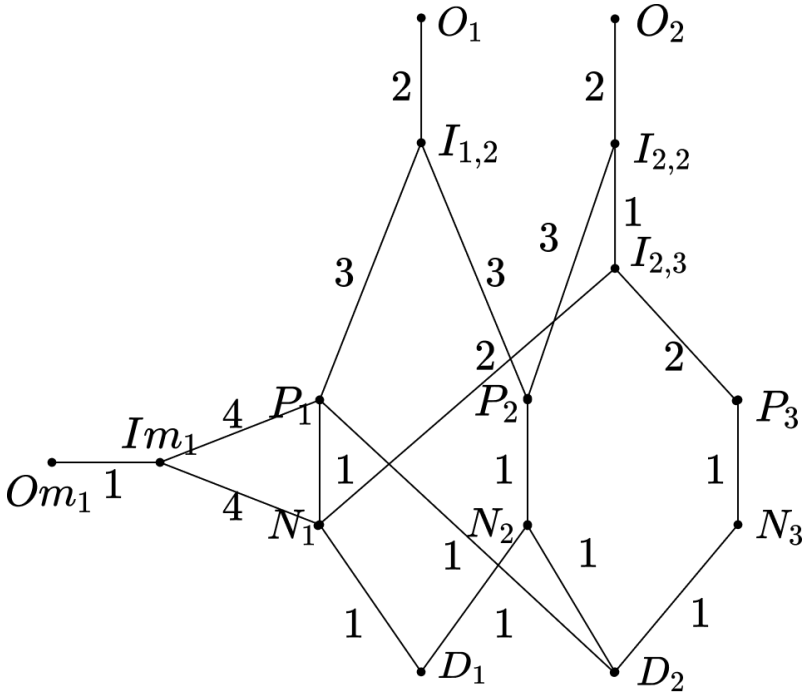

Figura 3.1: Grafo de la reducción sin modificar

Como aumentar el tiempo límite es inevitable, pero queremos que los objetos tengan exactamente un camino por cada literal, hemos de evitar este problema de alguna manera. Esta consiste en asegurarnos que la longitud de cualquier camino alternativo sea al menos n unidades mayor que la longitud de los caminos deseados (en el ejemplo anterior, los caminos deseados del objeto 1 son $O_1 - I_{1,2} - P_1 - N_1 - D_1$ y $O_1 - I_{1,2} - P_2 - N_2 - D_1$). Esto podemos lograrlo aumentando las distancias de las aristas que pueden ser utilizadas en caminos alternativos.

En la reducción actual, las aristas que van a los destinos tienen longitud 1. Podemos cambiar este número por n, y ya no se podrán tomar caminos que utilicen varias de estas aristas. Consecuentemente tenemos que añadirle otras $n - 1$ unidades al tiempo límite, ya que los caminos normales pasan por una de ellas. También pueden

tener longitudes menores a n las aristas que conectan algún vértice P o N con algún vértice I. Debido a esto, un objeto podría ir a un vértice intersección de otro objeto y continuar un camino alternativo hacia su destino. Para evitar esto, podemos añadir $n-1$ a todas estas aristas. Esto también nos hace aumentar el tiempo límite, pero este va a tener más cambios, así que podemos dejar el valor exacto del tiempo límite para el final. Adicionalmente, nos conviene también aumentar en $n-1$ la distancia de los vértices Im_i a P_i y N_i o los obstáculos llegarían antes que los objetos a dichos vértices.

Los cambios hasta ahora mencionados causan que la reducción no lleve al grafo de la figura 3.1 sino al de la figura 3.2.

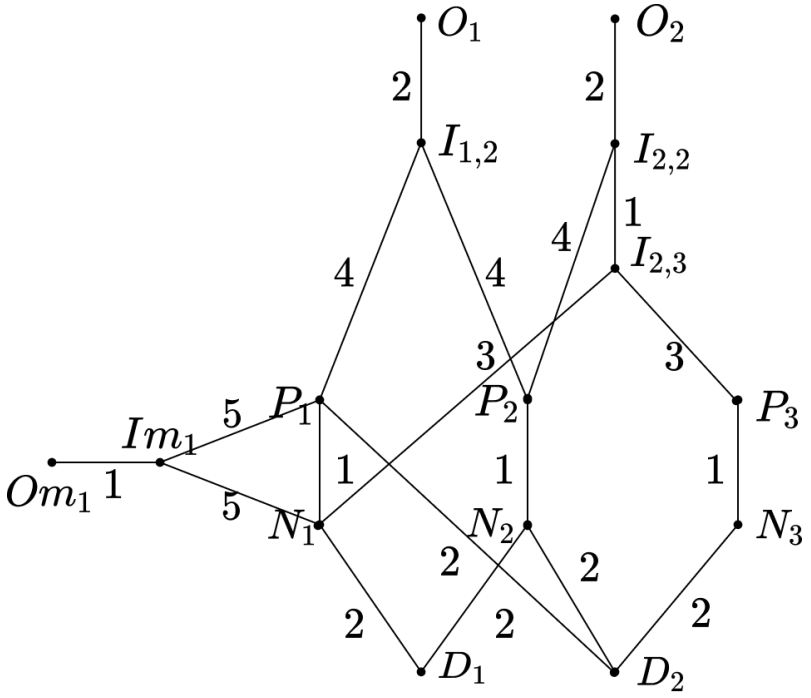

Figura 3.2: Grafo de la reducción con algunas modificaciones

Al haber colisiones en vértices, son también un problema los obstáculos. En primer lugar, en MnOmO cuando llegaban a un vértice P o N a la vez que un objeto no podían entrar en colisión con él. Sin embargo, en MnOmO-vértices esto sí provoca una colisión. Por tanto, para que nuestra reducción funcione en MnOmO-vértices, debemos retrasar la llegada del obstáculo en 1 momento. De esta manera sigue colisionando con el objeto si el obstáculo va a P y el objeto a N (ya que el objeto después tendrá que pasar por P) y viceversa, pero no colisionan cuando van ambos al mismo vértice.

Desgraciadamente, hay otros dos problemas relacionados con los obstáculos, causados por el hecho de que los objetos pueden tener que esperarse mutuamente para utilizar la misma arista. El primero de ellos es simplemente que algunos objetos pueden llegar hasta $n-1$ momentos después que el primero, por lo que habría que retrasar al obstáculo otros $n-1$ momentos. Esto causa que no colisione con la mayoría de objetos con los que sí debería colisionar, porque llega demasiado tarde, pero

se puede solucionar haciendo que las aristas $P - N$ tengan longitud n en lugar de 1. De esta manera, si el obstáculo va primero a P y después a N, colisiona con todos los objetos que recorran esa arista en el sentido $N - P$ pero con ninguno que la recorra en el sentido $P - N$.

El segundo problema surge cuando los obstáculos deciden detenerse. En la reducción a MnOmO, cuando un obstáculo se detenía podíamos ignorarlo justificadamente, ya que dejaba de poder chocarse con ningún objeto. En la reducción a MnO-vértices que estamos creando, algún objeto puede retrasarse hasta $n - 1$ momentos, por lo que si el obstáculo se retrasa esa cantidad de tiempo o menos, todavía no se puede ignorar. Esto es un inconveniente, porque si se detiene en Om o Im no sabemos a dónde va a dirigirse, pero tenemos que tomar decisiones como si lo supiéramos (tal y como en la demostración del lema 2.2.1).

Para poder saber a dónde se dirige el obstáculo o si podemos ignorarlo, tenemos que tomar nuestra decisión n momentos después de que el obstáculo llegue a Im, mientras que en la reducción original podía suceder que se tomaran decisiones tan solo 1 momento después. No pueden simplemente detenerse todos los objetos involucrados, ya que el resto de obstáculos podían seguir moviéndose. Por tanto, para solucionar este problema hemos de multiplicar todas las distancias por n. De esta manera se mantienen todas las proporciones, y si el obstáculo sigue detenido cuando los objetos lleguen a un vértice donde tienen que tomar una decisión, es porque lleva n momentos parado y puede ser ignorado.

Tras solucionar estos dos problemas, hemos realizado todos los cambios necesarios para que la reducción de TQBF a MnOmO-vértices sea equivalente a TQBF. Podemos ver en la figura 3.3 cómo queda el grafo al reducir la QBF 3.2. Se puede apreciar al comparar las figuras 3.1 y 3.3, que antes de las modificaciones los objetos y obstáculos llegaban a las intersecciones cada 1 momento y ahora cada 2 (ya que $n = 2$).

Se pueden ver las distancias de las aristas en la reducción de TQBF a MnOmO-vértices en la figura 3.3. Los vértices son los mismos que en la reducción a TQBF y también las funciones o, d, o_m que indican los orígenes y destinos. El tiempo límite ahora es $t = n * (k + 2 + 2n) = n * k + 2n^2 + 2$.

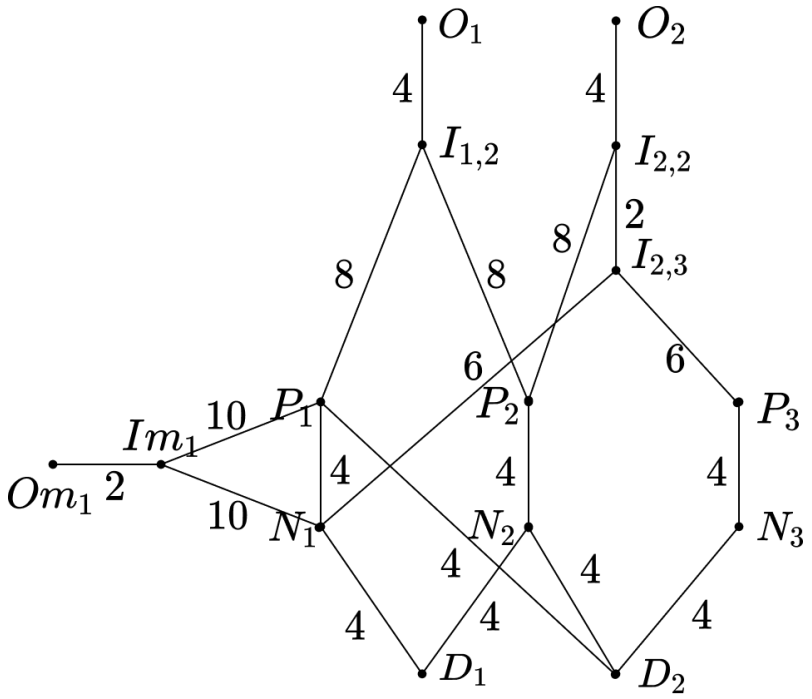

Figura 3.3: Grafo de la reducción con todas las modificaciones

$$
\begin{aligned}
(P_i, N_i, n^2) &\quad \forall i \in \{1, 2, \ldots, k\} \\
(Om_i, Im_i, n * i) &\quad \forall i \in M \\
(Im_i, P_i, n * (1 + k - i + n)) &\quad \forall i \in M \\
(Im_i, N_i, n * (1 + k - i + n)) &\quad \forall i \in M \\
(O_i, I_{i,j}, n * j) &\quad \forall i \in \{1, 2, \ldots, n\}, j = min(E_i) \\
(I_{i,j}, I_{i,s(j)}, n * (s(j) - j)) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \backslash max(E_i) \\
(I_{i,j}, P_j, n * (1 + k - j + n)) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \text{ t.q. } x_j \in C_i \\
(I_{i,j}, N_j, n * (1 + k - j + n)) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \in E_i \text{ t.q. } \neg x_j \in C_i \\
(I_{i,j}, P_l, n * (1 + k - j + n)) &\quad \forall i \in \{1, 2, \ldots, n\}, j = max(E_i), \forall l \in M \text{ t.q. } x_l \in C_i, l < j \\
(I_{i,j}, N_l, n * (1 + k - j + n)) &\quad \forall i \in \{1, 2, \ldots, n\}, j = max(E_i), \forall l \in M \text{ t.q. } \neg x_l \in C_i, l < j \\
(N_j, D_i, n^2) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \text{ t.q. } x_j \in C_i \\
(P_j, D_i, n^2) &\quad \forall i \in \{1, 2, \ldots, n\}, \forall j \text{ t.q. } \neg x_j \in C_i \\
(I_{i,l}, If_i, n * (k - l)) &\quad \text{si } \exists x_j \in C_i \text{ ó } \exists \neg x_j \in C_i \text{ t.q. } j > l \text{ donde } l = max(E_i) \\
(If_i, P_l, n^2) \in A &\quad \forall i \in \{1, 2, \ldots, n\}, \forall l \in M \text{ t.q. } x_l \in C_i, l > max(E_i) \\
(If_i, N_l, n^2) \in A &\quad \forall i \in \{1, 2, \ldots, n\}, \forall l \in M \text{ t.q. } \neg x_l \in C_i, l > max(E_i)
\end{aligned}
\tag{3.3}
$$

Con esto, queda definida una reducción de TQBF a MnOmO-vérices que se comporta igual que la de TQBF a MnOmO del anterior capítulo. Esto no quiere decir que sea la única reducción posible, ni necesariamente la más sencilla. Su ventaja es que al diseñarla a partir de la otra y arreglar los problemas individualmente, sabemos que las demostraciones que hicimos para la anterior reducción son análogas en esta.

Utilizamos este hecho para mostrar que MnOmO-vértices es un problema PSPACE-duro.

Lema 3.2.1. *Si una instancia de TQBF tiene respuesta afirmativa, entonces su reducción a MnOmO-vértices también también.*

Demostración. Análoga a la demostración del lema 2.2.1. □

Lema 3.2.2. *Si una instancia de TQBF tiene respuesta negativa, entonces su reducción a MnOmO también.*

Demostración. Análoga a la demostración del lema 2.2.2. □

Teorema 3.2.1. *MnOmO-vértices es PSPACE-duro.*

Demostración. El teorema 3.2.1 es consecuencia de los lemas 3.2.2 y 3.2.1, de que la reducción descrita es polinómica y de que TQBF es PSPACE-duro.

La reducción ha variado respecto a la anterior, aunque solo en la distancia de los vértices y en el tiempo límite. Estos han aumentado de manera polinómica, por lo que la reducción sigue siendo polinómica por las mismas razones que las expuestas en la demostración del teorema 2.2.1. □

Nos podemos también preguntar si puede ser más abordable que en el problema original la cuestión de si esta variante pertenece a PSPACE. Por desgracia, el árbol que representa todas las posibles combinaciones sigue siendo igual de inabordable. Sin embargo, el hecho de que un objeto no pueda nunca permitir que un obstáculo llegue a su mismo vértice hace que quizá sea más prometedora la estrategia de "podar" ramas inviables donde ya no es posible que los objetos lleguen a su destino sin riesgos.

3.3. MnOmO-destinos

Por último, vamos a tratar la variante que surge de quitar una de las reglas más aparentemente arbitrarias de MnOmO: la número 3. Proporcionamos alguna idea par ver que puede ser un problema NP-completo.

Llamamos MnOmO-destinos a la variante de MnOmO que no tiene la regla número 3. A continuación, recordamos cuál es dicha regla:

3. Los obstáculos no pueden comenzar un movimiento desde un vértice v_1 a otro v_2 si v_2 pertenece a $Im(d)$, donde $Im(d)$ es la imagen de la función $d : \{1, \ldots, n\} \to V$.

Como se puede ver, se trata de la regla que impide a los obstáculos ir a los destinos de los objetos. Por tanto, en MnOmO-destinos, los obstáculos sí pueden ir a los destinos. Como ya mencionamos en el anterior capítulo, parecería que el quitar esta restricción hace que sea más difícil el problema, puesto que al tener más opciones los obstáculos, el conjunto de simulaciones posibles distintas crece. Sin embargo,

creemos que esta variante pertenece a NP, propiedad que parece ser mutuamente excluyente con ser PSPACE-duro (es mutuamente excluyente si y solo si NP $\subsetneq$ PSPACE).

En primer lugar, hemos de resaltar que, sea PSPACE-dura o no, la variante MnOmO-destinos es NP-dura. Esto lo sabemos porque el mismo problema sin obstáculos ya es NP-duro (Carreño López, 2024), y opcionalmente podemos no añadir obstáculos a MnOmO-destinos para que sea equivalente.

Ahora, explicamos brevemente el por qué sospechamos que puede pertenecer a NP. Creemos esto, ya que parece que la mejor estrategia para los obstáculos es ir directamente a los destinos, lo cual reduce el número de casos que hay que considerar.

En general, si un obstáculo entra en colisión con un objeto en una arista $A - B$, con el obstáculo yendo de A a B y el objeto de B a A, entonces era posible que el obstáculo llegara al destino de ese objeto antes que el objeto, si no el obstáculo no toma la arista $A - B$ y se dirige hacia el destino por el camino más corto. Una vez un obstáculo está en el destino de un objeto, es imposible que dicho objeto llegue a él sin riesgo de colisiones. Por esto parece que la mejor estrategia para los objetos es ir directamente a algún destino. Por ejemplo, en los grafos creados por la reducción de TQBF a MnOmO, si los obstáculos van a algún destino por el camino más corto posible, en muchos casos pueden llegar 1 momento antes que el objeto correspondiente, haciendo negativa la instancia de MnOmO-destinos aunque la de TQBF sea afirmativa. Debido a esto no podemos adaptar dicha reducción para demostrar que MnOmO es PSPACE-duro.

Sin embargo, no está claro a qué destino deben dirigirse los obstáculos, ya que los objetos pueden seguir varios recorridos. Por ejemplo, si 2 objetos se tienen que esperar mutuamente para utilizar una arista en sentidos contrarios, puede llegar uno antes o el otro antes, dependiendo de cuál pase por la arista primero. Podría cada obstáculo ir al destino más cercano a ellos, pero muchas veces esta no será una estrategia óptima, como por ejemplo si hay menos obstáculos que objetos. Otra opción es que todos los objetos vayan al mismo destino, analizar qué deben hacer los objetos para llegar sin riesgos en ese caso, y repetir para cada destino. Sin embargo, en una simulación real es posible que si los objetos comienzan a dirigirse hacia un destino, los obstáculos reaccionen dinámicamente, luego los obstáculos reaccionen dinámicamente, y así indefinidamente. Debido a esto, no podemos afirmar que MnOmO-destinos pertenezca a NP, y es posible que sea NP-duro, pero es otra variante de gran interés.

Capítulo 4

Conclusiones y Trabajo Futuro

Hemos estudiado el problema de mover simultáneamente varios objetos evitando obstáculos en un grafo valorado. Hemos llegado a la conclusión de que es un problema PSPACE-duro y hemos discutido cómo podría llegarse a ver que pertenece a PSPACE. Posteriormente, hemos aprovechado estos resultados y la manera de llegar a ellos para estudiar varias variantes de dicho problema. Hemos mostrado que algunas variantes son PSPACE-duras y algunas de ellas PSPACE-completas. Al hacer esto, hemos visto que la misma reducción desde TQBF utilizada para mostrar que MnOmO es PSPACE-completa puede adaptarse para problemas similares. Para aquellos resultados que no hemos demostrado pero creemos que podrían ser ciertos, hemos dado algunas indicaciones de cómo se podrían abordar.

Una línea de trabajo futuro clara es tratar de avanzar con estos resultados que no se han podido demostrar. Por ejemplo, el ver si MnOmO y MnOmO-vértices están o no en PSPACE, con lo que se podría concluir si son PSPACE-completos o no. Otros resultados interesantes son los que conciernen a la clasificación de la complejidad de MnOmO-destinos, para los cuales seguramente sea importante estudiar a fondo las propiedades de dicho problema.

Por otro lado, se pueden tratar de investigar e implementar posibles algoritmos que intenten resolver aproximadamente el problema. Como ejemplo, se podría idear un algoritmo que trate de llevar a los objetos a su destino en simulaciones donde los objetos se mueven aleatoriamente. Esto podría intentarse con un algoritmo minimax que tras un par de niveles aplique como heurística cuánto tardarían en llegar los objetos a sus destinos desde su posición actual asumiendo que los obstáculos no se van a mover más. Para hallar el valor de esta heurística se podrían utilizar algoritmos ya existentes para este problema como los descritos en Alotaibi y Al-Rawi (2016), Standley y Korf (2011), Wagner y Choset (2015) y Ren et al. (2021). Con la misma idea de aproximar el problema, se podrían también tratar de adaptar a nuestro problema algoritmos ya existentes pensados para espacios euclídeos, como son los descritos en Masehian y Katebi (2007) y Bilbeisi et al. (2015).

Introduction

The study of computational complexity is a field of great interest for many researchers. Year after year it is discovered to which complexity class more and more problems belong to, and yet it seems that we are still unaware of some details about said classes' relationships. The truth is that it is a field that seems to always have more and more things to discover; this is what motivated us to realize this work, which studies the complexity of a small group of problems.

With these problems we refer to those regarding deciding whether it is possible to move a set of objects along a graph without risk to collide with a set of mobile obstacles in that same graph. We focus in one problem in particular and afterwards we study some of its variants. Previous work has studied the complexity of similar problems but without obstacles, which are NP-hard (Goldreich, 2011), (Carreño López, 2024). If we consider problems with more differences besides the lack of obstacles, we find a great quantity of similar problems, some NP-complete Ajay et al. (2022) and others PSPACE-complete (Hearn y Demaine, 2005). However, the complexity of the problems that will be studied in this work has not yet been analyzed, so we believe they are of theoretical interest.

These problems also are of practical interest, since they could model from strategies in a videogame to trajectories of drones that have to avoid real obstacles. Precisely due to this, how to approximate the movement of objects avoiding obstacles has already been studied in euclidean spaces (Masehian y Katebi, 2007), (Bilbeisi et al., 2015); and we hope that this work serves as inspiration so this is also studied in more generalized spaces such as graphs.

In chapter 2, we begin defining the main problem we are going to address. After that, we will use the PSPACE-complete problem *True Quantified Boolean Formula* (TQBF) to show that it is a PSPACE-hard problem. Subsequently, we study how its belonging to PSPACE could be proven. In chapter 3, we study some variants of these problems, proving some of their complexity properties. Finally, in chapter 4, we summarize the results and we give some ideas of future work.

As a work plan, in the first place, we have searched and read various publications of similar works. Afterwards, we have though about what properties our problem has and which ones we could prove. There we devised the reduction from TQBF

that has allowed us to prove that the problem is PSPACE-hard. We had to continue refining this reduction to be able to prove it produces instances equivalent to the originals. Following that, we thought of possible ways of proving that the problem is in PSPACE. When we realized none of them was going to be used in time for this work, we limited ourselves to compile the most promising ones for possible future works.

During this process, we noted the variants of the problem that seemed more promising to also study them. With them, we also though what we could prove and whether we could use the previous proofs to help us. Finally, we compiled all this work in this document including explanations and examples so it can be easily understood.

Conclusions and Future Work

We have studied the problem of moving simultaneously multiple objects while avoiding obstacles in a weighted graph. We have arrived to the conclusion that it is a PSPACE-hard problem and we have discussed how it could be proven that it is in PSPACE. Afterwards, we have used these results and the way to get to them to study a few variants of said problem. We have shown that some are PSPACE-hard and that some are even PSPACE-complete. When doing this, we have shown that the same reduction from TQBF used to show that MnOmO is PSPACE-complete can be adapted for similar problems. For the results that we haven't proven but we believe migh be true, we have given some indications about how to try to tackle them.

A clear line of future work is to try to progress with these results that are not yet proven. For example, whether MnOmO and MnOmO-vértices are in PSPACE, which would determine whether they are PSPACE-complete. Other interesting results are those concerning the complexity classification of MnOmO-destinos, for which it is probably interesting to study in depth the problem's properties.

A diferent line of future work is to investigate and implement possible algorithms that try to approximate solutions to the problem. For example, an algorithm that attempts to carry the objects to their destinations while the obstacles move randomly could be devised. One of the ways this could be done is with a minimax algorithm that, after a bit of exploration, applies as a heuristic how much the objects will take to get to their destinations assuming the obstacles will no longer move. To calculate the value of this heuristic, already existing algorithms could be used, such as the ones described in Alotaibi y Al-Rawi (2016), Standley y Korf (2011), Wagner y Choset (2015) and Ren et al. (2021). With the same goal of approximating the problem, it could be attemped to adapt existing algorithms designed for euclidean spaces such as Masehian y Katebi (2007) and Bilbeisi et al. (2015).

Bibliografía

Ajay, J., Jana, S. y Roy, S. Collision-free routing problem with restricted l-path. *Discrete Applied Mathematics*, vol. 319, páginas 71–80, 2022.

Allis, L. V. Searching for solutions in games and artificial intelligence. *Doctoral Thesis, Maastricht University*, 1994.

Alotaibi, E. T. S. y Al-Rawi, H. Push and spin: A complete multi-robot path planning algorithm. En *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, páginas 1–8. IEEE, 2016.

Bilbeisi, G., Al-Madi, N. y Awad, F. PSO-AG: A multi-robot path planning and obstacle avoidance algorithm. En *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, páginas 1–6. IEEE, 2015.

Carreño López, D. *Estudio del problema de la planificación de objetos móviles sin colisiones*. Bachelor thesis, Universidad Complutense, 2024.

Garey, M. R. y Johnson, D. S. *Computers and intractability*, vol. 174. freeman San Francisco, 1979.

Goldreich, O. *Studies in Complexity and Cryptography*, capítulo 1. Springer, Berlin, Heidelberg, 2011.

Hearn, R. A. y Demaine, E. D. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, vol. 343(1-2), páginas 72–96, 2005.

Masehian, E. y Katebi, Y. Robot motion planning in dynamic environments with moving obstacles and target. *International Journal of Computer and Information Engineering*, vol. 1(5), páginas 1249–1254, 2007.

Ren, Z., Rathinam, S. y Choset, H. Subdimensional expansion for multi-objective multi-agent path finding. *IEEE Robotics and Automation Letters*, vol. 6(4), páginas 7153–7160, 2021.

Schaefer, T. J. Complexity of decision problems based on finite two-person perfect-information games. En *Proceedings of the eighth annual ACM symposium on Theory of computing*, páginas 41–49. 1976.

Standley, T. y Korf, R. Complete algorithms for cooperative pathfinding problems. En *IJCAI*, páginas 668–673. Citeseer, 2011.

Wagner, G. y Choset, H. Subdimensional expansion for multirobot path planning. *Artificial intelligence*, vol. 219, páginas 1–24, 2015.